**UNIVERSIDADE FEDERAL DE MINAS GERAIS**
**Instituto de Ciências Exatas**
**Programa de Pós-Graduação em Ciência da Computação**

Giovanna Ávila Riqueti

# SAXJS: SAX BASED ONLINE CHANGE POINT DETECTION

Belo Horizonte
2024

Giovanna Ávila Riqueti

# SAXJS: SAX BASED ONLINE CHANGE POINT DETECTION

**Final Version**

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Heitor Soares Ramos Filho
Co-Advisor: Felipe Domingos da Cunha

Belo Horizonte
2024

Riqueti, Giovanna Ávila.

R594s SAXJS: [recurso eletrônico] SAX based online change point detection / Giovanna Ávila Riqueti - 2023.

1 recurso online (90 f. il., color.) : pdf.

Orientador: Heitor Soares Ramos Filho.
Coorientador: Felipe Domingos da Cunha.
Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciências da Computação.
Referências: f. 78-85

1. Computação – Teses. 2. Algoritmos de computador – Teses. 3. Análise de séries temporais - Teses. 4. Detecção de ponto de mudança - Teses. I. Ramos Filho, Heitor Soares. II. Cunha, Felipe Domingos da. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. IV. Título.

CDU 519.6*52(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx


UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# FOLHA DE APROVAÇÃO

## SAXJS: SAX BASED ONLINE CHANGE POINT DETECTION

## GIOVANNA ÁVILA RIQUETI

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Documento assinado digitalmente
gov.br HEITOR SOARES RAMOS FILHO
Data: 05/02/2024 17:03:12-0300
Verifique em https://validar.iti.gov.br

PROF. HEITOR SOARES RAMOS FILHO - Orientador
Departamento de Ciência da Computação - UFMG

Documento assinado digitalmente
gov.br FELIPE DOMINGOS DA CUNHA
Data: 08/02/2024 11:37:47-0300
Verifique em https://validar.iti.gov.br

PROF. FELIPE DOMINGOS DA CUNHA - Coorientado
Instituto de Ciências Exatas e Informática - PUC Minas

Documento assinado digitalmente
gov.br JOAO BATISTA BORGES NETO
Data: 06/02/2024 09:29:01-0300
Verifique em https://validar.iti.gov.br

PROF. JOÃO BATISTA BORGES NETO
Departamento de Computação e Tecnologia - UFRN

Documento assinado digitalmente
gov.br MARCELO AZEVEDO COSTA
Data: 06/02/2024 07:42:08-0300
Verifique em https://validar.iti.gov.br

Prof. MARCELO AZEVEDO COSTA
Departamento de Engenharia de Produção - UFMG

Documento assinado digitalmente
gov.br FREDERICO GADELHA GUIMARAES
Data: 06/02/2024 10:31:03-0300
Verifique em https://validar.iti.gov.br

PROF. FREDERICO GADELHA GUIMARÃES
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 5 de fevereiro de 2024.

*I dedicate this work to the people who helped me do it: Heitor, Felipe, Pedro and João. I also dedicate it to my family, friends and boyfriend who gave me all the support I needed.*

# Acknowledgments

Firstly, I would like to thank my family, especially my parents and my sister, Leda, Luis and Julia, who have always been very patient with me, always encouraged me to study and helped me with everything I needed on this journey, even if what I needed was just a friendly hug. I am also very grateful to my boyfriend, Gabriel, who was with me in all the moments of happiness and sadness in this very difficult process that is the master's degree, and my friends Olivia, Amanda and Giovanna who listened to my worries when I most needed.

I would like to express my deep gratitude to my advisors, Heitor Ramos, Felipe Domingos, as well as my colleagues Pedro Barros and João Borges, whose professionalism has always impressed me. Even when they were very busy, they were always helpful and showed me research paths that I had not thought of before. They taught me to be more patient and to fight for my goals, lessons that I will carry with me for the rest of my life. A special thanks to Felipe, who has been on this journey with me for the longest time and has always believed in my potential since graduation and opened my path to multiple professional opportunities.

My sincere thanks to my professors from PUC Minas, Luis Zárate and Max do Val Machado for showing me how to be a great computer scientist and always challenging me to do my best. Also, I would like to thanks DCC - UFMG for providing an excellent academic structures that enable the development of this work with excellent professors, staff and students. I am grateful for everything I learned at UFMG, whether talking to other students or in the classroom with professors.

"*A person who never made a mistake never tried anything new.*"
(Einstein, Albert)

# Resumo

Séries temporais são representações organizadas cronologicamente ao longo do tempo, usadas para descrever a dinâmica de um determinado sistema. O comportamento deste sistema pode apresentar variações ao longo do tempo, e os pontos no tempo onde essas variações ocorrem são chamados de pontos de mudança. Detectar esses pontos de mudança é uma tarefa importante com diversas aplicações relevantes para o mundo atual, como no monitoramento de mudanças climáticas, investigação de problemas médicos, compreensão de ondas cerebrais, padrões de sono e aprimoramento da classificação de imagens como uma etapa de pré-processamento.

Apesar da detecção de ponto de mudança ser uma área explorada na literatura por décadas, ainda existem alguns desafios. Notavelmente, existe a demanda por algoritmos que operem de forma *online* sem que seja preciso a informação da série temporal por completo, e por algoritmos que demonstrem adaptabilidade a diversos sistemas. Neste trabalho, nós propomos um novo método capaz de resolver esses desafios. Nossa abordagem é baseada em transformações de séries temporais do tipo *Symbolic Aggregate ApproXimation* (SAX), simbologia de *Bag-of-Patterns* (BOP), probabilidade de transição entre símbolos e distância Jensen-Shannon. Avaliamos nosso algoritmo proposto utilizando casos sintéticos e dados do mundo real, comparando-o com métodos de detecção de pontos de mudança do estado da arte e obtivemos os melhores resultados em aproximadamente 71% dos experimentos.

**Palavras-chave:** detecção de ponto de mudança; representação de séries temporais; *Symbolic Aggregate Approximation* (SAX).

# Abstract

Time series are chronologically organized representations over time used to describe the dynamics of a given system. The behavior of this system can manifest variations over time, and the points in time where these variations occur are called change points. Detecting these change points is an important task with several relevant applications for the current world, such as monitoring climate change, investigating medical issues, understanding brainwaves, sleep patterns, and enhancing image classification as a pre-processing step.

Despite being an area explored in the literature for decades, the change point detection field still has some challenges. Notably, there are demands for online algorithms capable of operating without the entirety of time series information, and algorithms that demonstrate adaptability across diverse systems. In this work, we propose a novel framework to solve these challenges. This framework is based on Symbolic Aggregate ApproXimation (SAX) transformation, Bag-of-Patterns (BOP) symbology, the probability of transition between symbols, and the Jensen-Shannon distance. We evaluate our framework with synthetic data and real-world cases in comparison to state-of-the-art change point detection algorithms and we have the best values in approximately 71% of the experiments.

**Keywords:** change point detection; time series representation; Symbolic Aggregate Approximation (SAX).

# List of Figures

# List of Tables

# List of Algorithms

# List of Symbols

| | |
|---|---|
| $\alpha_i$ | represents the SAX symbol which is the i th element of the alphabet. |
| $\beta$ | If its to use frequency domain in TIRE implementation. |
| $\delta$ | If its to use time domain in TIRE implementation. |
| $\hat{C}$ | A sequence of SAX symbols. |
| $\mathcal{P}_t, P_{t+W}$ | Probability distribution of two sequential windows. |
| $\sigma$ | Controls the relative PE-divergence used in RuLSIF implementation. The lower the $\sigma$, the more the density ratio is plain and less "smooth". |
| $\tau$ | Number of the next neighbor to establish the connection between SAX symbols. |
| $a_{ij}$ | count of the transitions between two SAX symbols in nodes $i$ and $j$ in a transitional graph. |
| $b$ | A user-defined number of SAX symbols. |
| $d$ | Number of neighbors to concatenate a SAX symbol. |
| $f_{b^d}$ | relative frequency of a BOP symbol. |
| $f_b$ | relative frequency of a determinated SAX symbol. |
| $g$ | Number of neighbors to fit the low-degree polynomial in the Savitzky–Golay filter application. |
| $Gamma\ \alpha_0$ | $\alpha$ parameter of the Inverse Gamma used in BOCPDMS. |
| $Gamma\ \beta_0$ | $\alpha$ parameter of the Inverse Gamma used in BOCPDMS. |
| $h$ | Number of features ($timeinvariant + instantaneous$) used in TIRE. |
| $I$ | Tolerance interval to consider a change point. |
| $K$ | Number of kernels in RuLSIF method. |
| $Ka$ | Number of Autoencoders to use in TIRE implementation. |
| $M$ | Length of Fourier transform used in TIRE implementation. |

| | |
|---|---|
| $N$ | Time series size. |
| $n_{al}$ | Number of all change points alarms. |
| $n_{cc}$ | Number of times the change point alarm was correct. |
| $n_{FP}$ | Number of false positives. |
| $n_{rc}$ | Number of all change points in a time series. |
| $p$ | Number of neighbors to compare in the peak detection. |
| $S$ | Time series with size $N$. |
| $s_t$ | A point in time series $S$ in timestamp $t$. |
| $t$ | A timestamp in a time series. |
| $W$ | Window size. |
| $w_0, w_1$ | Time series sequence of two consecutive windows. |
| $X$ | Bag-of-pattern(BOP) aggregation with the SAX symbols and its $d$ neighbors. |

# Contents

# Chapter 1

# Introduction

The current chapter introduces this dissertation work entitled "SAXJS: SAX Based Online Change Point Detection". We discuss this work's initial considerations and motivation in Section 1.1. After, in Section 1.2, we show the goals we intend to achieve. Section 1.3 highlights the contributions made. Finally, Section 1.4 describes the organization we followed in this work.

## 1.1 Motivation

Time series are an ordered sequence of data evenly spaced over time used to describe the behavior of a given system. This system's behavior may exhibit changes, such as the rise in temperature captured by a sensor near a fire. These changes can have external causes, like a fire outbreak, or internal causes, such as a sensor malfunction.

Detecting these change points (CPs) is an important task with several relevant applications for the current world, such as detecting climate change to monitor the effects of global warming [62], investigating some medical problems, as well as understanding brainwaves [23] and sleep monitoring [6], for example. It also can be applied to improve image segmentation and, consequently, image classification [75].

Another application is detecting a point in the field of human activity using wearable sensors. The detection of change points is helpful in this application to segment activities, minimize interruptions while interacting with humans, provide activity-aware services, and help monitor health status [2].

The algorithms to detect these changes can be divided into two main fields: offline and online. The offline techniques are only applicable with the entire time series information [67]. The online, otherwise, can be real-time or $\varepsilon$-real-time, where the $\varepsilon$ is the number of points necessary to look ahead [2]. The need for more online algorithms is one of the most crucial challenges in the study area of change point detection.

Despite being an area explored in the literature for decades [5], it still has challenges

other than the need for online algorithms. For instance, there is a necessity for algorithms that can be adaptable in different systems. In addition, in many real-world problems, the amount of change is more interesting than the change itself. Another example, climate researchers can be interested in knowing how much the temperature changes instead of just detecting the change [2].

In order to solve these challenges, our work proposes a new online framework for detecting change points, called SAXJS, based on comparing the probability distribution of time series symbolic representation. The time series symbolic representation chosen was the Symbolic Aggregate ApproXimation (SAX) [42] transformation, and the distance to measure probability distributions was the Jensen-Shannon distance [44]. Our algorithm has four different versions, which are SAXJS-Symbols (SAXJS-S), SAXJS-Transitional (SAXJS-T), SAXJS-$\tau$ and SAXJS-BOP. In all versions, we transform the data of two adjacent sliding windows into an SAX representation and calculate a probability distribution. This probability distribution is computed considering the probability of a symbol or the probability of symbols' transition depending on the version approach.

In both the SAXJS-S and SAXJS-BOP versions, we consider symbols's probability distribution. However, in SAXJS-S we consider the SAX representation as the symbols, and in the SAXJS-BOP we use the Bag-of-Patterns (BOP) [43] symbology with a user-defined number of neighbors. In SAXJS-T and SAXJS-$\tau$, we create a SAX symbols' transition graph and consider each edge probability. The difference between SAXJS-T and SAXJS-$\tau$ is that SAXJS-$\tau$ adds a $\tau$ hyperparameter. Unlike SAXJS-T, where the symbol connects to its next neighbor in time $t$, in SAXJS-$\tau$, it connects to its neighbor in time $t + \tau$.

In all versions, the windows' probability distribution is compared by calculating the Jensen-Shannon distance between them. For each Jensen-Shannon value, we applied the Savitzky–Golay filter [52], which fits a low-degree polynomial based on its adjacent data to smooth the data. As the Jensen-Shannon value is smoothed we also apply a maximum value detection algorithm that analyzes a user-determined number of neighbors. The maximum value must be greater than a defined threshold value to be considered as a possible change point.

## 1.2 Objectives

The main objective of this dissertation is to answer the following question:

**Research problem**: Is it possible to detect change points in an online and adaptable to different systems manner using time series symbolic transformation

probability distribution?

To that end, a fundamental step is to understand how a time series can be transformed into a symbolic representation and how this symbolic representation can improve our goal of detecting a change point. Thus, we tackle the main objectives of this dissertation by answering the following questions:

- **How can we extract change point information from a time series?** Our goal here is to use the time series symbolic transformation in the design of a framework to identify the breakpoints in a time series in which the behavior is changed. Our method must extract time series features that, efficiently and effectively, help to identify the change points.
- **Is SAX symbols' distribution capable of providing good change point detection?** In the design of our framework, one of our hypotheses is that transforming the time series into a SAX symbols distribution based on the characteristics of this method (such as computationally inexpensive and robustness to outliers) can create a satisfactory online change point detection. Positively contributing to this field.
- **Is SAX symbols' transition distribution capable of providing good change point detection?** In the design of our framework, another hypothesis is that transforming the time series into a transition distribution of SAX symbols can provide a more robust online change point detection as the transition information can be of great value to detect frequency changes.
- **Is Bag-of-Patterns (BOP) based distribution capable of providing good change point detection?** Another hypothesis is that establishing a symbology by integrating the concept of Bag-of-Patterns (BOP) into SAX symbols could enhance robust online change point detection. This enhancement stems from the superior capability of Bag-of-Patterns (BOP) effectively representing the connections between neighborhoods compared to isolated SAX symbols, which can lead to better frequency changes detection.

## 1.3 Contributions

Our main contributions can be summarized as follows.

- **Transition graph with SAX symbols:** We propose a novel time series representation based on the transitional graph of Symbolic Aggregate Approximation (SAX) symbols, providing better change point results.
- **Bag-of-Patterns (BOP) with sliding window division at the beginning of the process:** We proposed a novel time series representation based on sliding windows at the beginning of Bag-of-Patterns(BOP), providing better change point results.
- **Definition, modeling, and application of Online change point Detection Framework:** We proposed a novel online change point detection framework with four variations, using SAX transformation, transitional graph, and BOP transformation.

## 1.4 Outline

The remainder of this dissertation is organized as follows. Chapter 2 provides a literature overview about time series symbolical representation and change point detection algorithms. Chapter 3 presents our proposal of a novel change point detection called SAXJS and highlights its advantages and disadvantages. Chapter 4 describes the parameter configurations, the data sets, and the metrics used to evaluate our framework. Chapter 5 compares our framework proposal with other state-of-art change point algorithms and evaluate our method in change point free environment. Chapter 6 summarizes the contributions of this dissertation and presents some research directions for our future works.

# Chapter 2

# Literature Review

This work presents a novel online change point detection framework that compares a time series symbolic representation probability distribution called SAX. As this work involves time series symbolic representations and change-point detection, in this chapter, we discuss different time series symbolic representations and highlight the advantages of SAX for our proposal of constructing an online change-point detection method in Section 2.1. Section 2.2 discusses distinct approaches to detect change points. Section 2.3 presents our final remarks, finalizing the chapter.

## 2.1 Time Series symbolical representations

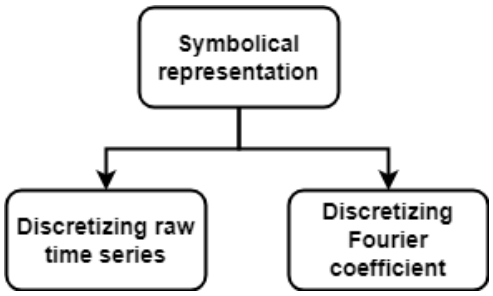
A time series is a collection of chronological observations of one or many dimensions. One of its most distinct characteristics is its high dimensionality. Although analyzing the whole series can be important for its statistics, it is not required for many data mining algorithms, as their goal is more centered on knowledge discovery. Therefore, it is necessary to find a time series representation that can reduce the time series dimensionality while preserving its essential characteristic to the analyses. Reducing the time series dimensionality is also important to lower execution time [17] and reducing storage space since fewer bits are required for each value [41]. Although there are many time series representations, we focus on symbolic transformation to create a probability distribution of symbols to measure the similarity between distributions and identify change points.

The symbolic transformation consists of discretizing a time series numerical value into symbols [16]. This approach opens the possibility of using algorithms and data structures associated with symbolical representation, for example, hashing, Markov models, suffix trees, decision trees, and the Jaccard coefficient. Utilizing a symbolic data structure can aid in constructing a probability distribution, as it reduces dimensionality, it allows for a controlled number of symbols in a time series, and enhancing robustness to noise by representing noisy data with symbols. Moreover, the chosen methodology for transforming

time series values into symbols can accentuate crucial features for change point detection, enhancing the sensitivity of symbol distribution to factors such as changes in amplitude.

We found in the literature two two approaches for symbolic transformation of time series: one based on discretizing raw time series and the other based on discretizing Fourier coefficients [19] as shown in Figure 2.1. We present these two approaches in Sections 2.1.1 and 2.1.2, respectively, and discuss the advantages and disadvantages of each symbolic representation. In Section 2.1.3, we discuss which transformation method is more effective in achieving our goal of extracting useful information to detect change points.

Figure 2.1: Diagram of symbolic transformation.




Source: Created by the author.

### 2.1.1 Approaches based on discretizing raw time series

The most commonly used transformation to discretize raw time series is the Symbolic Aggregate Approximation (SAX). The SAX technique [41] involves normalizing the time series data to achieve a mean ($\mu$) of zero and unitary variance, empirically ensuring an equiprobable distribution of symbols [54]. Having an equiprobable distribution, SAX constructs a Gaussian curve and determines the breakpoints that divide the time series into $b$ equal-sized areas under a Gaussian distribution curve. Each area under the Gaussian curve corresponds to a symbol. SAX has a low computation cost of $O(n)$, and symbols occupy less space than numbers. The SAX is commonly used with Piecewise Aggregate Approximation (PAA) [32]; first, the time series is reduced to PAA representation, and then the SAX is used. PAA is a dimensionality reduction method representing each fixed-length segment as its mean value.

Besides its low computational cost and better space occupation, the SAX transformation also allows distance measures to be defined under symbolic space as it lower bounds the true distance between the original time series. This lower bound property also ensures that the algorithm's assertiveness remains unaffected, even in cases where the time series deviates from a Gaussian distribution [41]. Additionally, it also enhances

robustness against noise by converting noisy data into standardized symbols according to their distribution within the Gaussian curve.This characteristic is important to provide satisfactory results to data mining analysis, such as clustering, classification, and anomaly detection [41]. Nevertheless, SAX has the disadvantage of not capturing certain features or behaviors of the time series, such as trends, the amplitude distance between SAX symbols, and the frequency of SAX sequences and sets of symbols.

There are several SAX variants proposed in the literature, and part of them works to add information to the SAX symbols in the sequence, capturing these distinct features or behaviors that SAX does not initially capture. This type of variation is discussed in Section 2.1.1.1. The other part of algorithms based on SAX uses SAX sequence structures to create a new representation. This other strategy is mentioned in Section 2.1.1.2.

#### 2.1.1.1 Strategy based on adding information to SAX symbols

One of the first SAX variants [46] consists of adding trend information by calculating the minimum and maximum of each segmentation. However, this approach triples the SAX transformation dimension. The TD-SAX [64] and SAX-BD [25] also try adding trend information to SAX symbols differently. TD-SAX [64] calculates the distance between the endpoints of the segments and doubles the SAX symbol's dimension. SAX-BD [25] also doubles the SAX symbol's dimension by adding the boundary distance, which is the distance from the minimum and maximum value to the average value.

To study the spread of the values within a segment and improve the similarity measure calculation, the SAX-SD [78] adds the standard deviation information. EN-SAX [79] tries to solve a different problem than its ancestors. Its goal is to overcome the SAX drawback of two-time series with different shapes that may be mapped to the same SAX representation. To solve this problem, EN-SAX adds the entropy information of the series, which can measure the degree of disorder in the series. SAX-VFD [76] also uses entropy; however, it uses entropy in a vector with various other information such as statistical measures of mean, standard deviation, kurtosis, and skewness. It also includes fluctuation features such as slope, absolute energy, and sum over first-order differencing. Each feature is represented as an SAX segmentation.

Multivariate Symbolic Aggregate Approximation (MSAX) [49] is a SAX variant to represent multivariate time series. First, MSAX checks if the variables are independent of each other. If they are, the data are normalized. If they are dependent, a linear transformation is applied. Then, the PAA and SAX transformations are applied, and the matrix is transformed into a sequence of symbols.

All of the SAX variants discussed in this section, with MSAX as an exception, increase the dimensionality of the symbolical representation. Increasing the dimensionality escalates memory consumption and increases the necessity for a larger time series to ensure a reasonable sample size for all symbols. Therefore, using these types of SAX variants is not an advantage to our algorithm's focus on probability distribution unless it adds an essential feature to detect change points. EN-SAX introduces a compelling feature for change point detection, enabling the identification of time series patterns, which can be valuable for detecting shifts in frequency. Nonetheless, it adds a real value to the symbolic representation, making the probability calculation complex.

#### 2.1.1.2 Strategy based on using SAX sequence structure to create a new representation

Instead of adding information to the SAX representation, other algorithms use the SAX sequence structure to create a new representation. One of the first methods that use this approach is the Bag-of-patterns [43]. The Bag-of-patterns (BOP) involves concatenating the SAX symbols with their neighboring symbols using sliding windows. For instance, consider a time series sequence denoted as $\hat{C} = \hat{c}_1, \cdots, \hat{c}_N$ with size $N$, consisting of $b$ symbols options, with a chosen value of $d = 3$ to represent the number of neighbors to be concatenated. For example, having a symbol $c_t$ in timestamp $t$ will be concatenated with $c_{t+1}$ and $c_{t+2}$ becoming one single symbol. In creating these subsequences, in some cases, we might see that multiple consecutive subsequences are mapped to the exact string. Therefore, to avoid over-counting these sequences, there is an option to apply numerosity reduction, which consists of recording only the first sequence associated with its first appearance and ignoring the rest until we find another sequence. As an example, suppose we obtain the following sequence of SAX strings:

$$\hat{X} \equiv aba\ aba\ aac\ aac\ aac\ cab\ cab$$

With the numerous reductions, we would record the following sequence instead:

$$\hat{X1} \equiv aba_1 aac_2 cab_5$$

Having this sequence of symbols for the entire time series, we divide this sequence into sliding windows to create the $M_{ij}$ matrix, where $i$ is the number of symbols and $j$ is the first timestamp of each sliding window. Each $M(i, j)$ value is the frequency of the $ith$ subsequence occurrence in the $jth$ sliding window.

The BOP method encounters a memory occupancy challenge as the matrix size grows exponentially with the combination of symbol variations ($b$) and the number of neighboring symbols to concatenate ($d$). For example, when $b = 4$, you have four SAX symbol options (a, b, c, d), and with $d = 5$, you generate $4^5 = 1025$ rows in the matrix. Despite its apparent size, the matrix is likely to be sparse, as is the case with text data. In experiments, they find that only about 10% of all strings have some subsequence assigned to them. One way to overcome this problem is to eliminate words that never occur in the data or to set a maximum row threshold size.

The BOP strategy of grouping sequences of SAX symbols can be interesting for the problem of detecting change points since using the sequence as a symbol it would be possible to capture the oscillations of the data, thus capturing changes in variance and frequency. To capture variance and frequency changes, in this work, we will use the BOP to create a new symbology whose probability assignment will be used to identify change points.

Based on BOP, another algorithm called Symbolic Aggregation approXimation in Vector Space Model (SAX-VSM) [61]. The SAX-VSM has two significant differences from BOP. The first difference is that the time series is separated into sliding windows at the beginning of the process instead of at the end. The other difference is that SAX-VSM applies TF-IDF, a technique to evaluate the importance of a word in a document, using each sliding window as a document to create a frequency matrix. Using the TF-IDF method, the number of rows becomes the only subsequence that appears in a time series. Therefore, the matrix size does not grow exponentially. Nevertheless, employing TF-IDF is not a suitable approach for our proposal. This is because, to construct a probability distribution from the symbols in two consecutive windows, we must ensure that the symbols are consistent between both windows. This requirement is not met when only the occurring symbols are considered, as a symbol may be present in one window but absent in another.

One of the most recent methods that use SAX sequence structure to create a new representation is the Pattern-Based Embedding for Time Series Classification (PETSC) [20]. First, PETSC transforms a time series into SAX transformations using sliding windows. Then, to discover sequential patterns, they extend techniques from frequent discriminative and top $k$ pattern mining to reduce the number of candidate sequences [18, 22]. The best candidate sequences vary in length, duration, and resolution. These sequences are then used to train a linear model. PETSC has the advantage of having low computational costs and supports real-time applications. PETSC also does not require stringent assumptions about the time series format, making it more adaptable to data from different contexts. However, varying the sequence in length and duration can make its sequence repetition harder to reproduce a probability distribution.

### 2.1.2 Methods based on discretizing Fourier coefficients

Instead of discretizing raw time series, these methods apply Fourier coefficient-based transformations and then discretize to obtain symbolic information. The most commonly used method in this field is the Symbolic Fourier Approximation (SFA) [58]. The SFA algorithm consists of two steps. Discrete Fourier Transformation (DFT) [71] is applied in the first step. The idea of DFT is to decompose a time series using sinusoidal waves as basic functions. Each wave is represented by a complex number $X_f$ identified as the Fourier coefficient, where the magnitude represents the amplitude and the phase is the initial phase of the angle. The Fourier coefficients are transformed into symbols in the second step using the Multiple Coefficient Binning (MCB) transformation. MCB transformation divides the coefficients into $w$ groups. Then, it applies an equi-depth or equi-width binning to each histogram to define the $c$ numbers of breakpoints, defining the intervals corresponding to an alphabetic symbol.

Based on the SFA transformation, the Bag-of-SFA symbols (BOSS) [57] algorithm divides a time series into sliding windows in each applied SFA transformation. All SFA symbols in all sliding windows are counted to construct a frequency vector. The numerosity reduction is often used to avoid outweighing subsequences. BOSS reduces the SFA complexity from $O(W\ log\ W)$ to $O(1)$, where $W$ is the window size, and is more invariant to noise. There are multiple versions of BOSS; one example is the Randomized BOSS (RBOSS) [47]. RBOSS introduces a random factor in the choice of the sliding window lengths, which makes it easier for the classifier to build and save memory space. Another example is the SP-BOSS [37], which combines temporal and phase-independent features and uses Spatial Pyramids and other computer vision strategies.

Similarly to BOSS, word extraction for time series classification (WEASEL) [59] also uses sliding windows and SFA transformation in each window. The distinction lies in WEASEL's utilization of multiple window sizes and the ANOVA test to pinpoint optimal breakpoints for binning Fourier Coefficients, enhancing its efficacy in tackling classification problems. Consequently, WEASEL relies on a supervised step for this purpose. Furthermore, WEASEL incorporates a feature selection process that uses the Chi-square test to reduce the size of the feature vector while retaining critical information essential for classification. WEASEL plus Multivariate Unsupervised Symbols and Derivates (MUSE) [60] is its multivariate version.

The Temporal Dictionary Ensemble (TDE) [37] combines the Spacial Pyramids Algorithm (BOSS-SP) and the original BOSS to create a symbology. This method has the same problem as BOSS-SP, which is the high memory cost of Spatial Pyramids. As the Fourier Coefficient transforms the series into the frequency domain, the symbolical transformation only captures the frequency aspect of the change. Therefore, using a

discretization applied to the Fourier coefficient is not an interesting idea for our project, and we chose to try different strategies to capture frequency changes.

### 2.1.3 Discussion

As shown in this section, many options exist for symbolic discretization of time series in the literature. However, few meet the objective of our work, which is to build a change point algorithm based on distribution probability. Constructing a probability distribution necessitates a limited set of symbols, as an excess of symbols may lead to some not appearing within a segment of the time series, demanding an extensive time series length. The incorporation of real numbers to the SAX symbolization in most cases amplifies the number of potential symbols significantly, making these symbolizations impractical.

Furthermore, many symbolic representations transform time series into Fourier coefficients and then apply the symbolic transformation. This process highlights changes in the frequency range to the detriment of identifying changes in amplitude and variance. This more restricted approach would not help our algorithm, whose main characteristic is that it is adaptable to different systems. Another issue contributing to some representations, such as PETSC, that our algorithm cannot use is that they have a variable symbol size. When there are several symbol possibilities, it becomes more difficult to calculate the probability of each symbol, as we do not know which one symbol will appear for a given time series.

Therefore, we decided to adapt SAX and BOP to achieve our goals and create a change point to capture different changes. The advantages of SAX are its low computational cost, its lower bound, its robustness to noise and the capture of amplitude shift in time series [41]. This possibility of capturing amplitude shift is a standout compared to other discretization methods based on equi-width and equi-depth. Despite these advantages, SAX symbolization has disadvantages, as its symbolic representation does not capture the variance and frequency shifts. To fix it, we propose two versions of our framework (SAXJS-T and SAXJS-$\tau$) to use SAX with graphs, as it is possible to see the shifts in connection patterns instead of symbols. By identifying patterns of connections between symbols, it is easier to identify data oscillations and, consequently, changes in frequency and variance.

In another version of our framework called SAXJS-BOP, we use a BOP variation to capture the connection of symbols as it makes a new representation by concatenating SAX symbols in a sequence. The drawback of SAX-BOP is its exponential growth of symbols,

which leads to the necessity of time series with more length to turn its frequency matrix less sparse [43]. We detailed our framework and how we adapted SAX and BOP to solve the change point problem in Chapter 3.

## 2.2 Change point detection

Change point detection (CPD) is the problem of detecting abrupt and smooth changes in a time series. Still, differently from anomaly detection, the underlying dynamics of the time series changes [2]. Through the last decade, many algorithms have been designed and adapted to solve the CPD problem. In Section 2.2.1, we review the techniques used in the algorithms to detect change points. Section 2.2.2 defines the criterion used to compare CPD algorithms based on its characteristics.

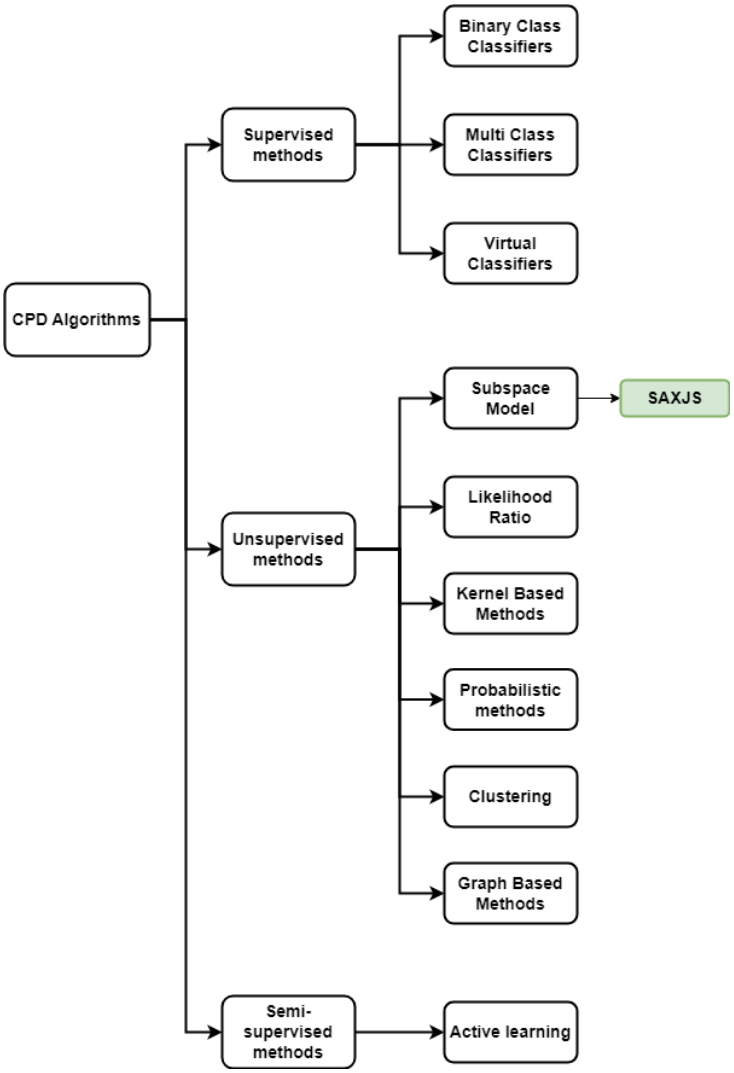
### 2.2.1 Algorithms techniques

Figure 2.2 presents a taxonomy of the techniques used for change point detection. The first division in the taxonomy is the Supervised, Unsupervised, or Semi-Supervised methods.

#### 2.2.1.1 Supervised methods

The Supervised methods need manual labeling change point examples to train the model to predict the correct output for new input time series data. The **multi-class** classifiers are used when the number of possible process states is specified, and the change point algorithm is trained to find each state boundary. This approach uses a sliding window through the data, analyzing if each point is a change point and needs a diverse and sufficient amount of training data to learn each space's underlying characteristics and differentiate them. A variety of machine learning classifiers can be used in Multi-Class strategies such as the Gaussian mixture model (GMM) [12], Support Vector Machine [53] and Hidden Markov model [24].

Figure 2.2: Diagram of change point detection methods.




Source: Strongly based on [3].

The **Binary class** problems only recognize two states: if the point is a change point or not, which can lead to a class imbalance problem. Some machine learning classifiers such as Support Vector Machine, Naive Bayes, and logistic regression [21] were tested using this strategy. The Pyramid Recurrent Neural Networks for Multi-Scale Change point Detection (PRNN) [15] is also a supervised learning method that is designed to be scale-invariant by incorporating wavelets and pyramid analysis on Neural Network.

The third Supervised methods approach is the **Virtual Classifier** [26], which can also be used to interpret the change between two consecutive windows. For each consecutive window, the first window is attached with the hypothetical label (+1) and the second with (-1). Then, The algorithm is trained with any human-interpretable machine learning method such as Association Rules and Decision Tree. Next, the algorithm changes the label of the ones classified as change points, increasing its accuracy.

#### 2.2.1.2 Unsupervised methods

Conversely, unsupervised learning (UL) strives to find patterns or relationships in a data set without being given any labeled examples. UL segments the data in a change point detection scenario and analyzes its statistical structure [3]. **Subspace Model** is a UL paradigm where states represent the time series, and to detect change points, these approaches estimate the distance between these state spaces. For example, the Subspace Identification (SI) algorithm [31] builds an observation matrix based on the state space model generated through a sliding window. Based on this matrix, the authors calculate the distances between the subspaces to detect change points. The Singular Spectrum Transformation (SST) [48] is also a Subspace Model algorithm. The major difference between SI and SST is SI considers a noise factor, and SST does not consider the noise system.

Besides Singular Value Decomposition (SVD) used in SI and SST, it is also possible to use an autoencoder to create this subspace, as done in Autoencoder-based Breakpoint. Detection procedure (ABD) [39]. The Partially time-invariant representation (TIRE) [14] uses an autoencoder to estimate time-invariant features. But instead of only using the time-invariant features, the authors use the Fourier transform to extract frequency domain information and combine both (time and frequency). TIRE uses the Euclidean distance between the sliding window to detect changes in the dynamics. Lastly, ADAptative GAussian (ADAGA) [7] is also an example of the subspace approach to detect change points. ADAGA infers a Gaussian Process model for a window and a subwindow and uses a hypothesis test to evaluate if the subwindow data is from a different Gaussian process. One of its main strengths is the adaptability of the subwindow and window sizes as they increase until the change point, and then the window is spoiled.

Another UL paradigm utilizes **Likelihood Ratio** to analyze two consecutive intervals of the same state based on their probability density or probability distribution comparison. The first step of this approach is to calculate the probability density of the intervals separately, and the second step is to compute the ratio between the probabilities. The most known Likelihood Ratio algorithm for change point detection is the Cumulative Sum (CUSUM) [51]. CUSUM accumulates the likelihood of deviations to a reference point, and when its accumulated sum exceeds a threshold, the approach detects a change point. Change Finder [74] is another commonly used method whose strategy consists of fitting a time series onto an Auto Regression (AR) model to get its statistical behavior. The difference of the AR model probability distributions of consecutive spaces is calculated. The AR model is applied again to the result time series difference, and a score function is calculated. CUSUM and Change Finder rely on pre-designed parametric models and are less flexible in real-world scenarios.

Unlike the predecessors that calculate both consecutive intervals probability densities, Kullback-Leibler (KL) importance estimation procedure (KLIEP) [63] focuses on the direct ratio of probabilities. KLIEP estimates the direct ratio using KL divergence, which is defined by the equation comparing probability $p(x)$ and probability $p'(x)$:

$$KL[p(x)||p'(x)] := -\int p'(x) log \frac{p(x)}{p'(x)} dx. \tag{2.1}$$

The method Unconstrained Least-squares Importance Fitting (uLSIF) [27, 30] also estimates the direct ratio, but using Pearson (PE) divergence defined by the equation:

$$PE[p(x)||p'(x)] := -\int p'(x)(\frac{p(x)}{p'(x)} - 1)^{.2}$$

KL and PE divergence can be unbounded depending on the condition of the second interval probability $p'(x)$. The Relative uLSIF (RuLSIF) [45] solves this problem by using an $\sigma$-relative value between $0 < \sigma < 1$ changing the PE equation as follows:

$$PE_\sigma[p(x)||p'(x)] := PE(p(x)||\sigma p(x) + (1-\sigma)p'(x)).$$

RuLSIF is bounded above $\frac{1}{\sigma}$, and it is proved that its convergence ratio is faster than uLSIF [73]. If $\sigma = 0$, the density ratio is reduced to a plain and gets "smoother" as $\sigma$ gets bigger.

**Kernel-based methods** uses unsupervised kernel-based test statistics to check the homogeneity of data in two windows. As one example of this strategy, KernelCPD [8, 4] uses one of the three Kernels: linear, cosine, or Gaussian, and is complemented with a Pruned Exact Linear Time (Pelt) [34]. Pelt detects change points by minimizing a cost function over possible locations and the number of change points. To reduce the computational cost, Pelt prunes the data set using the minimal distance between change points (a.k.a. *minsize*), a cost function regularizer (i.e., a larger regularized value induces the model to find more change points), and the grid of possible change points. More recently, studies have tried a deep learning approach. The Kernel Learning Change Point Detection (KL-CPD)[9] is a kernel learning framework that optimizes a lower bound of test power using an auxiliary generative model.

Following the **probabilistic methods**, which tries to estimate the probability based on previous probabilities, there is the Online Bayesian Network (OBN) [1]. OBN model learns a probability distribution based on the last "run length", the time since the most recent change point. This gives rise to a recursive message passing between the joint distribution of "run length" and observation data. OBN had some recent modifications as the Bayesian On-line Change point Detection with Model Selection (BOCPDMS) [35], which is proposed as a spacially structured Vector Autoregression (VARs) for modeling the space between change points and deciding online the best models for representing the data. The Robust BOCPDMS (RBOCPDMS) [36] adds the $\beta$-divergence to the General

Bayesian Inference, which gives different weights to observation data and reduces the influence of the outliers.

The change point problem can also be adapted to a **Clustering** problem by comparing if a point $t$ in time is in the different cluster as its next point $t+1$ and how far these clusters are. The Sliding Window and bottom-up (SWAB) [33] uses a bottom-up cluster strategy, and in every window, the leftmost cluster sequence is reported and replaced in the next window. In contrast, the Shapelet-based [77] attempts to separate the data by recursively clustering the time series based on its shape until no separation is possible. The last change point technique is the Unsupervised field is **Graph-based methods**. This technique is used in the work of Chen and Zang [10], in which the time series points are represented as graph nodes, and a two-sample test is applied in this new representation.

#### 2.2.1.3 Semi-supervised methods

To solve the Supervised learning problem of having a sufficient amount of annotated data, which may not always be available, and the UL problem of possibly making wrong assumptions in the parametric approach, the **Semi-supervised** field appears as a new possibility in the change point literature. The AL-CPD [13] implements this new approach by using Active Learning, which queries a human annotator to find labels that efficiently can better identify change points. First, the algorithm works unsupervised using TIRE to find candidates' change points. Then, a Random Forest Classifier is run as a form of double validation for the candidates. The TIRE method was adapted to a semi-supervised version to find new change points by changing the time-invariant loss closer to the change points confirmed by the Random Forest.

### 2.2.2 Criterion to classify Change Point Detection algorithms

In this Section, we present the two principal criterion used to analyze CPD algorithms according to its characteristics: whether the algorithm operates in an 'online' or 'offline' mode, and its adaptability to multiple systems. The offline techniques only apply with the complete time series information [67]. The online, otherwise, can be real-time or $\varepsilon$-real-time, where the $\varepsilon$ is the number of points necessary to look ahead [2]. Offline and

Online have different applicabilities. The first is more related to data segmentation, and as it has the whole time series information, the number of False positives can decrease. The online is great for monitoring systems, and it has the advantage of notifying the change point close to its occurrence in time, thus enabling faster actions to be taken [40]. Online algorithms are one of the most crucial challenges in the study area of change point detection.

The second criterion involves assessing the algorithm's adaptability to various systems. This adaptability is evaluated based on certain methodological characteristics. Firstly, it is determined whether the method learns in a Supervised or Unsupervised manner. As Supervised method depends on having a sufficient quality and quantity of training data [2] it is not applicable to multiple real-world scenarios, presenting a drawback to their adaptability across multiple systems.

The second characteristics is if the method is parametric or nonparametric. Parametric approaches specify a particular function form to be learned and try to estimate its unknown parameters. In contrast, the nonparametric does not make any assumptions about the properties of an underlying function [2]. As it makes fewer assumptions, the nonparametric algorithms are more adaptable to different systems with different data sizes. The nonparametric approach usually has better computational cost and scale compared to parametric algorithms [10].

One example of parametric approach is the Online Bayesian Network (OBN), which makes an assumption about the prior probability distribution. The newer versions of OBN, including BOCPDMS and RBOCPDMS smooths the prior probability distribution choice by using a Model Selection step, although the assumption regarding the prior probability distribution persists, maintaining a parametric approach. Although SAX assumes a normal distribution, its assertiveness remains unaffected even if this assumption is not fulfilled due to its inherent lower bound property, which is not the case of those using Bayesian statistical techniques. This feature renders SAX more adaptable to different systems than these methods.

## 2.3 Chapter Remarks

Change Point Detection (CPD) has many applications, an area explored in the literature for decades, as seen in this Section. However, it still has some challenges, such as the need for online methods to deal with stream data and more adaptable algorithms for different systems [2]. The table 2.1 defines CPD algorithms according to their technique, if they are online and if they have some drawbacks that may reduce its adaptability to

different systems or its scalability. As with most real-world problems, the change points are not known, in Table 2.1, we only compare and analyze Unsupervised techniques.

Table 2.1: CPD algorithm criteria.

| Method name | Algorithm technique | Online | Drawbacks |
|---|---|---|---|
| SI | Subspace model | ✓ | Parametric approach |
| SST | Subspace model | ✓ | Parametric approach. |
| **SAXJS** | Subspace model | ✓ | SAX makes assumption about the data following an Gaussian distribution. |
| ABD | Subspace model | ✓ | Difficult to choose hyper-parameters in complex models. |
| TIRE | Subspace model | ✗ | Need to know if the data has frequency changes before its application. |
| ADAGA | Subspace model | ✓ | The implementation of the algorithm has a scalability problem. |
| CUSUM | Likelihood Ratio | ✓ | Parametric approach |
| Change Finder | Likelihood Ratio | ✓ | Parametric approach |
| KLIEP | Likelihood Ratio | ✓ | Cross-validation step reduces scalability. |
| ulSIF | Likelihood Ratio | ✓ | Cross-validation step reduces scalability. |
| RulSIF | Likelihood Ratio | ✓* | Cross-validation step reduces scalability. |
| KernelCPD | Kernel-based methods | ✗ | Complexity of $O(n^2)$. |
| KL-CPD | Kernel-based methods | ✗ | Difficult to choose hyper-parameters in complex models. |
| OBN | Probabilistic methods | ✓ | Parametric approach. |
| BOCPDMS | Probabilistic methods | ✓ | Parametric approach. |
| RBOCPDMS | Probabilistic methods | ✓ | Parametric approach. |
| SWAB | Clustering | ✓ | It does not have many recent mentions in the literature. |
| Shapelet-based | Clustering | ✗ | It does not have many recent mentions in the literature. |
| Graph-based method | Graph-based method | ✓ | It does not have many recent mentions in the literature. |

* Adapted to online in this work.

In this work, we sought to validate our proposal by comparing it with methods employing various algorithmic techniques, giving preference to the most recent, online,

and adaptable approaches. We used TIRE as a benchmark for the subspace modeling technique, comparing it to SAXJS, as it incorporates more contemporary technologies such as autoencoders and deep learning. Given its online nature, we also attempted a comparison with ADAGA. However, we encountered a significant runtime issue, making completing the analysis on larger time series data impractical within our timeframe. For a single time series of substantial size, ADAGA's grid-search runtime exceeded five days, which led us to exclude the ADAGA analysis from our primary work. We could only generate results using grid-search for the Run Log data set as it is the smallest data we tested. We also could generate results using its default value for synthetic data sets with changes every 100 steps. Both results and the ADAGA's parameter configuration can be found in Appendix A. Nevertheless, we propose to include ADAGA analysis in future research. Despite ADAGA's use of techniques to reduce complexity, the use of Scipy library have a scalability issue if larger time series [7].

For the likelihood ratio, we choose to compare with RulSIF, which is more robust than ulSIF and KLIEP and a more computationally efficient method than KLIEP [45] and is a trendy method in literature [2]. Despite being less expensive than KLIEP, it still has a step of cross-validation, such as ulSIF, which also has the best $\sigma$ parameter for kernel estimation, which still hinders its implementation. RulSIF was originally offline because of its post-process of detecting change points based on a threshold. Therefore, we changed this post-process to transform online, but we kept the essence of density estimation that defines RuLSIF implementation. We perform the same post-process as SAXJS, smoothing the results and applying peak detection.

We tested with KernelCPD for Kernel-based methods. For Probabilistic Methods, we tested BOCPDMS because it is less complex than RBOCPDMS, and both are online [69]. The problem with probabilistic methods is that they have to make assumptions about the parameters of the data. BOCPDMS and RBOCPDMS try to make it more adaptable by performing an online model selection [35, 36].

The advantage of SAXJS over these online models is that it does not require a search for better parameters, such as RuLSIF. This results in reduced complexity and computational costs. Its computational cost is also better than KernelCPD and than algorithms that use more complex techniques such as autoencoders and deep learning. Moreover, SAXJS is more adaptable to different systems than BOCPDMS as it makes less assumptions about the data. Additionally, SAXJS does not need to know if the data have frequency changes to decide its hyperparameters as TIRE has. As Clustering and Graph-based techniques are not frequently cited in the recent literature, we decided not to include these two techniques in the analysis.

# Chapter 3

# SAXJS: our proposed method

In this chapter, we want to answer our first specific objective:

How can we extract change point information from a time series?

To answer this question and achieve our goal, we developed a framework composed of five steps: symbolic aggregation approximation (SAX) transformation, probability distribution construction, Jensen-Shannon distance calculation, and data smoothing and peak detection. Before we present our framework steps, we define and formalize the problem in Section 3.1. In Section 3.2, we explain the five steps of our proposed algorithm and highlight the difference between the four SAXJS variations (SAXJS-S, SAXJS-T, SAXJS-$\tau$ and SAXJS-BOP). In Section 3.3, we detailed the algorithm in a pseudo-code and analyzed its time complexity. In Section 3.4, we finalize the chapter discussing the advantages and disadvantages of each SAXJS variation.

## 3.1 Preliminaries, Fundamentals, and Definitions

Before we present our proposal, we will briefly define and formalize the problem. These definitions will be used throughout the thesis.

**Definition 1** *A time series is a sequence of data points indexed (or listed or represented graphically) by some time representation. More commonly, a time series is a sequence of successive points typically equally spaced in time; thus, it is a discrete-time data sequence.*

**Definition 2** *Let $\boldsymbol{S}$ be a time series with $s_t \in \boldsymbol{S}$ in a particular timestamp $t$ with size $N$. We assume the existence of some probability distribution function $\mathcal{P}$ for $s_t \in \boldsymbol{S}$. Therefore, with this definition, the time series can be considered a stochastic process [66]. Since $\mathcal{P}$ can be described by a set of parameters $\pi_t$ for a given point $s_t$, we can formalize that if there is an existence of a change point in the times series $\boldsymbol{S} = \{s_1, s_2, s_3, \cdots, s_N\}$ we have:*

$$\pi_{s_1} = \pi_{s_2} = \cdots = \pi_{s_T^*} \neq \pi_{s_{T^*+1}} = \cdots = \pi_{s_N},$$

*where we consider the change point between* $\pi_{T^*}$ *and* $\pi_{T^*+1}$ *because* $\pi_{T^*} \neq \pi_{T^*+1}$. *More formally,* $\exists T$ *where we have* $T^* \leq T < T^* + 1$, *if this inequality does not occur, we don't have a change point. We define* $T^*$ *as being the time of the change point of* $\boldsymbol{S}$ *in the interval* $\{s_1, s_2, s_3, \cdots, s_N\}$.

Detecting a change point is a challenging task as it tries to identify a change in the intrinsic characteristics of the data rather than an anomaly. Some approaches consider finding outliers (anomalies) and, by doing so, investigate how these outliers are related to the change points [65]. The change points are related to a change in the dynamics of the underlying model. So, change points have characteristics of a new model and can not be considered an outlier. Assuming a model change, detecting a change point can be seen as a change of dynamics (or phase transition) detection.

**Definition 3** *Let* $\mathcal{A}$ *be an algorithm that finds a change point* $T*$ *in the interval* $\{s_1, s_2, s_3, \cdots, s_n\}$ *for a time series* $\boldsymbol{S}$. *We define that* $\mathcal{A}$*-performance is given in* $\lambda$*-real time as the interval needed to detect the change point* $T$. *So, if an algorithm is said to be* $\lambda$*-real time we have it find the change point* $T$ *in the interval* $[t - \lambda, t + \lambda]$, *where* $t - \lambda \leq T < t + \lambda$. *The offline approach considers the whole time series* $S$ *having a* $\lambda$ *value approximated to* $N$, *as for the online approach, the* $\lambda$ *can be a smaller number compared to* $N$.

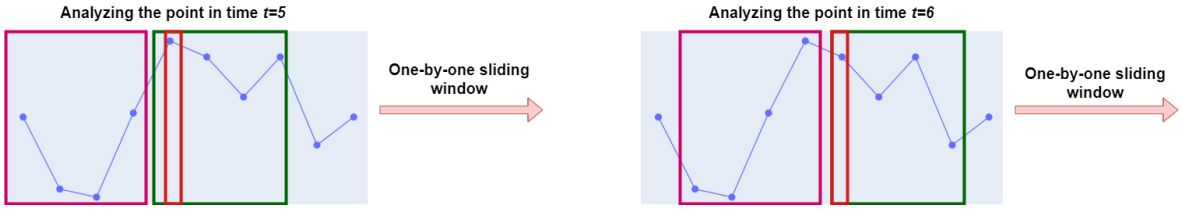
As an online framework, we define two adjacent windows that represents an slice of $S$ of size $W$. The first window is characterize by the time series $w_0 = \{s_0, s_1, s_2, ..., s_W\}$ and the second window is characterize by the time series $w_1 = \{s_{W+1}, s_{W+2}, s_{W+3}, ..., s_{2W}\}$. We discretize both windows into two sequences of symbols $\hat{C}_0 = \{\hat{c}_1, \cdots, \hat{c}_W\}$ and $\hat{C}_1 = \{\hat{c}_{W+1}, \cdots, \hat{c}_{2W}\}$. We calculate the distance between the probability distribution of both windows symbols sequences to evaluate if the point $W + 1 \in w_1$ is a change point. Then, we move to next point $W + 2$ and repeat the process.

## 3.2 SAXJS algorithm steps

This section presents our framework steps to detect if a certain point in time $t$ of a time series $S$ is a change point. As showed in Figure 3.1, our framework operate using two consecutive sliding windows of size $W$ and evaluate if the first point of the second

window is a change point or not. The Figure 3.2 summarize our framework steps and the operations made inside each window.

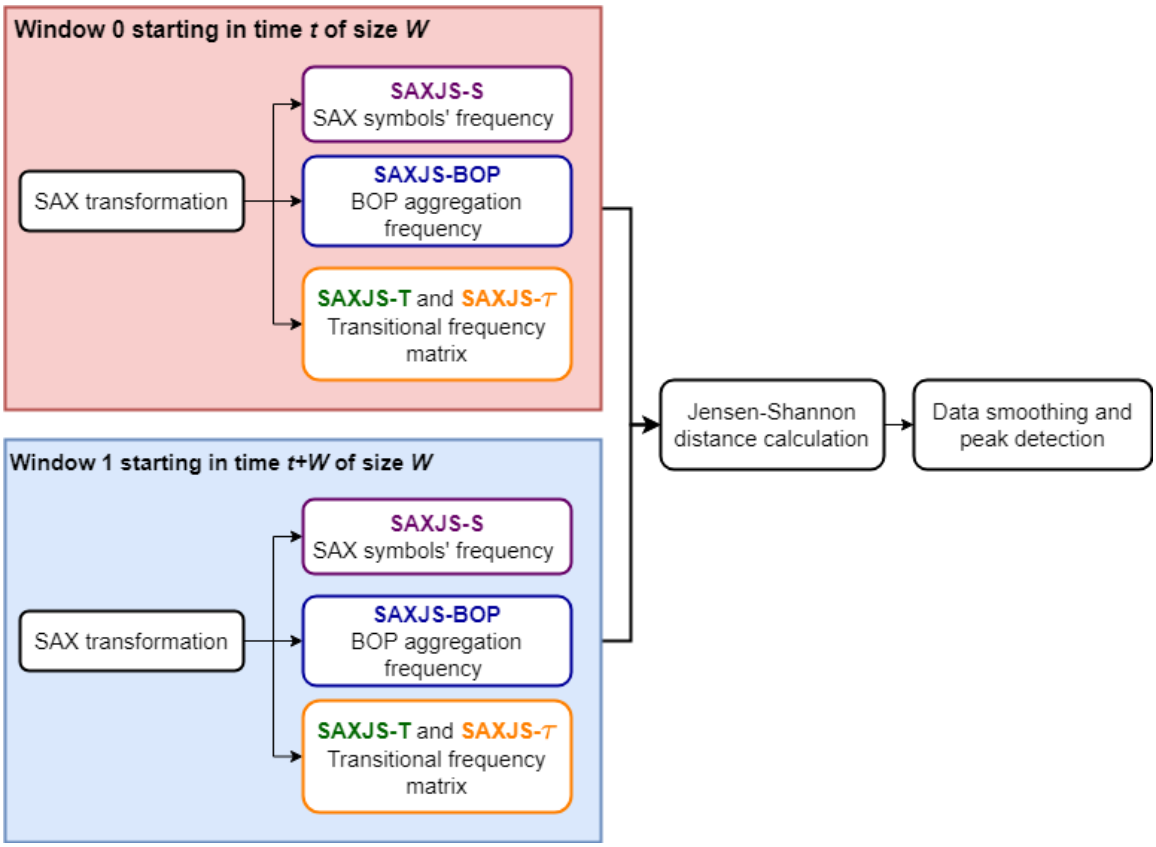
Figure 3.1: Example of sliding window operation.




Source: Created by the author.

In Section 3.2.1, we explain the first step of our algorithm: applying SAX transformation in our time series in both two adjacent windows. Section 3.2.2 describes how we extract the probability distribution from the SAX sequence. We use three distinct strategies to calculate the probability distribution, and each strategy corresponds to a different SAXJS variation. Section 3.2.3 presents the Jensen-Shannon distance calculation used to compare two probability distributions. In Section 3.2.4, we describe the data smoothing process of the Jensen-Shannon values and the peak detection method used to identify change points.

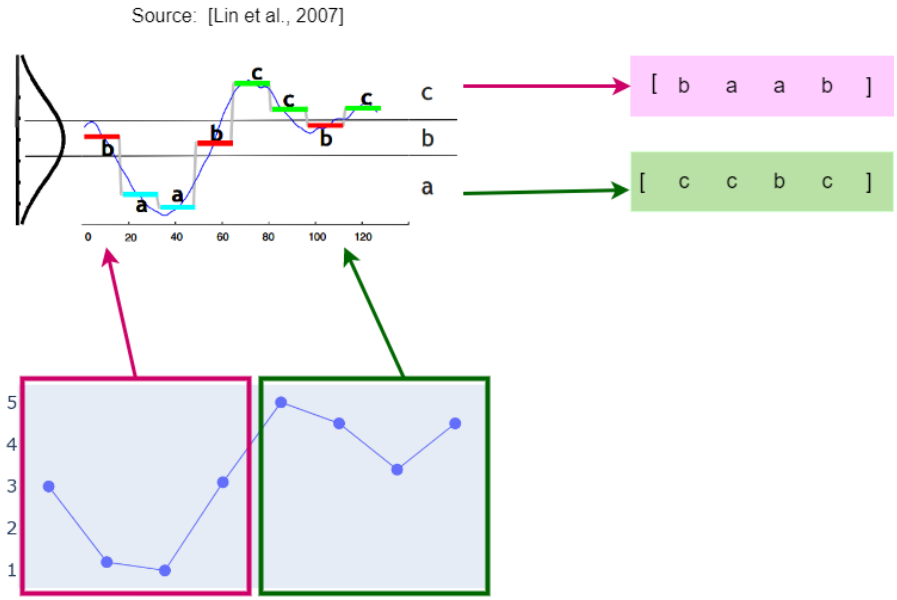
Figure 3.2: Diagram of our Framework steps using two adjacent windows of size $W$.




Source: Created by the author.

### 3.2.1 SAX transformation

Figure 3.3: SAX whole process using two sequential windows.




Source: Created by the author inspired by [42].

First, we discretize the time series windows $w_t$ and $w_{t+W}$ of size $W$ into a symbolic string sequence using the Symbolic Aggregate Approximation (SAX) transformation [42]. The SAX process with two sequential windows is shown in Figure 3.3. As evident from the Figure, the data from both windows are normalized and fit into a Gaussian distribution with three equal-sized areas. Points within the first area are denoted by the symbol $a$, those within the second area are represented as $b$, and the third area is designated as $c$. Upon completion of this process, we obtain the SAX sequence for each window, with the SAX sequence in pink corresponding to the first window and the green sequence corresponding to the second window.

The SAX algorithm considers that since a normalized time series has a Gaussian Distribution [38], we can determine the breakpoints that divide the time series into $b$ equal-sizes areas under the Gaussian curve, being $b$ a user-defined value. The breakpoints are a sorted list of numbers $B = \{\beta_1, ..., \beta_{\alpha-1}\}$ such that the area under a $\mathcal{N}(0, 1)$ Gaussian curve from $\beta_i$ to $\beta_{i+1} = \frac{1}{b}$ are all equal-sized.

Therefore, all the time series elements are represented by letters according to their value compared to the breakpoints. For example, if the value is under the smallest breakpoint, it is mapped to the symbol “a”.

Let $\alpha_i$ represents the $i^{th}$ element of the alphabet, for example, $\alpha_1 = a$ and $\alpha_3 = c$. Thus, considering the time series $w_0 \equiv \{s_t\}_{t=0}^{W}$ of size $W$ and time $t$, the mapping from

a time series representation to a word $\hat{C} = \{\hat{c}_1, \cdots, \hat{c}_W\}$ is obtained by $\hat{C}_i = \alpha_j$, if $\beta_{j-1} \leq \overline{c_i} < \beta_j$.

### 3.2.2 Probability distribution calculation

After we had the SAX symbolization with $b$ symbols, there are three possible strategies for calculating the probability distribution, thereby generating multiple versions of the SAXJS algorithm. In the following subsections, we describe these three possibilities. In Section 3.2.2.1, we describe the SAX symbols frequency calculation strategy, corresponding to the SAXJS-Symbols (SAXJS-S) version. In Section 3.2.2.2, we elaborated the transitional frequency matrix strategy on both SAXJS-Transitional (SAXJS-T) and SAXJS-$\tau$ versions. In Section 3.2.2.3, we explain the SAXJS-BOP variation (probability distribution based on Bag-of-patterns (BOP)).

#### 3.2.2.1 SAX symbols frequency (SAXJS-S version)

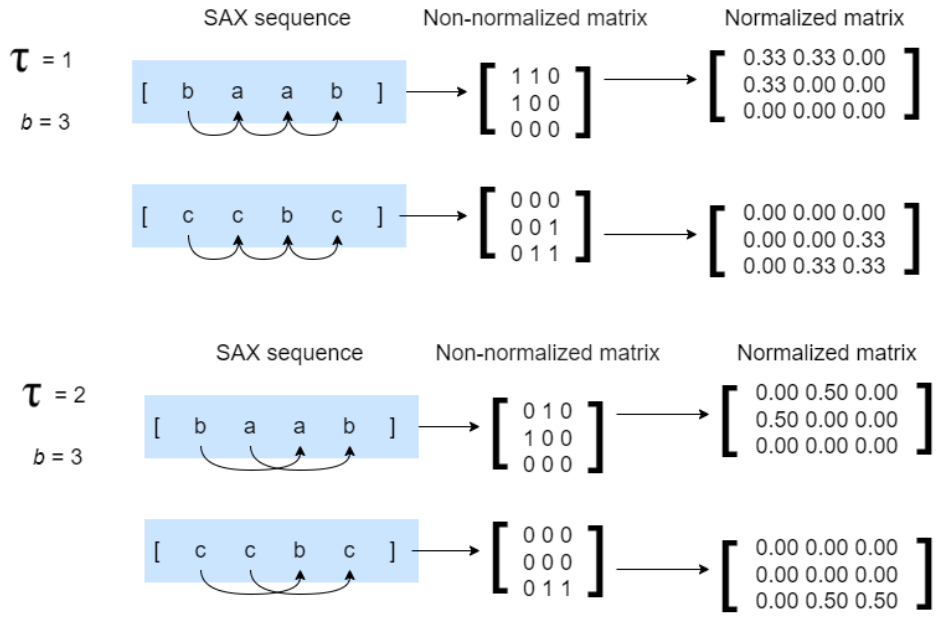
The first option to obtain the probability distribution is the SAXJS-S version, which involves calculating the relative frequencies of all $b$ symbols. This relative frequency can be computed when a particular symbol appears in a time series window ($f_b$), divided by the window length ($W$). As a result, calculating these relative frequencies allows us to construct the probability distribution histogram $\mathcal{P} \equiv p(b)$, which is defined by:

$$p(b) = \frac{f_b}{W}. \tag{3.1}$$

#### 3.2.2.2 Transitional Frequency Matrix (SAXJS-T and SAXJS-$\tau$ versions)

The second option consists of calculating the probability distribution based on a matrix of the transition frequency of each symbol. The process of constructing this matrix is shown in Figure 3.4 using a small window as an example. As shown in Figure, we create a matrix with each $i$ row and each $j$ column is an alphabetic symbol according to the

Figure 3.4: Example of constructing the symbols' transition frequency matrix and the effects using different $\tau$. The SAX sequences are the same use in Figure 3.3.



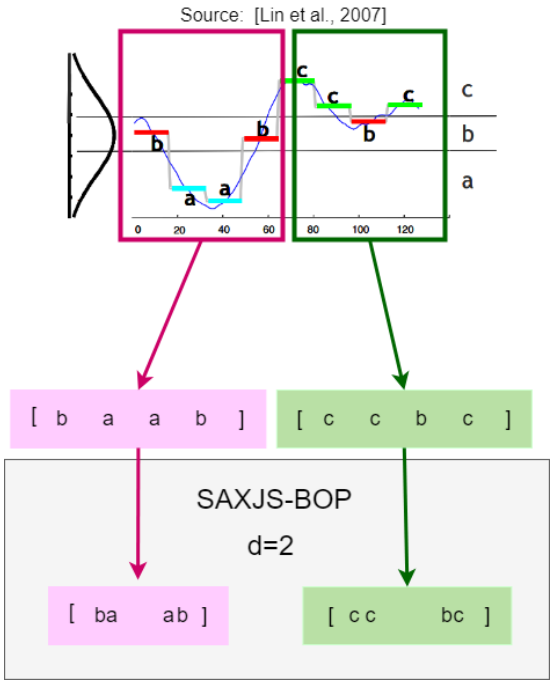

Source: Created by the author.

number $b$ of possible SAX symbols. Considering the SAX sequence $\hat{C} = \{\hat{c}_1, ..., \hat{c}_W\}$, we connect the symbol $c_t$ in time $t$ with the symbol $c_{t+\tau}$. The distinction between SAXJS-T and SAXJS-$\tau$ lies in the use of $\tau$: while SAXJS-T sets $\tau$ to a fixed value of one, SAXJS-$\tau$ allows for a variable $\tau$ value. Then, we count the sequential connections from one symbol $\alpha_t$ to another symbol $\alpha_{t+1}$ and add this count to the matrix in position of row $\alpha_t$ and column $\alpha_{t+1}$. This matrix arrangement resembles a directed graph, with each SAX symbol representing a node and the weight of each edge indicating the frequency of connections between symbols.

The matrix is converted to a probability matrix using Equation (3.2) being $a$ each count of transitions between the SAX symbols in each row $i$ and each $j$ column. Then, the probability matrix is converted to a 1-dimensional array by row order.

$$P_{a_{ij}} = \frac{a_{ij}}{\sum_{j=1,i=1}^{A} a_{ij}}. \tag{3.2}$$

Figure 3.5: Example of constructing the new symbolical sequence based on SAX sequences. The SAX sequences are the same use in Figure 3.3.




Source: Created by the author inspired by [42].

#### 3.2.2.3 SAX symbols aggregation based on Bag-of-Patterns (BOP) (SAXJS-BOP version)

The third option is to use the SAX symbols in an aggregation based on Bag-of-Patterns (BOP). Figure 3.5 illustrates the process in a small data set as an example. Taking into account the discrete SAX sequence $\hat{C} = \{\hat{c}_1, \cdots, \hat{c}_N\}$ of size $W$, we group the symbols in $\hat{C}$ into groups of user-defined size of $d$. For example, if we have $d = 3$, a symbol $c_t$ in time $t$ will be concatenated with symbols $c_{t+1}$ and $c_{t+2}$ generating one single symbol $X$ following the representation $X = c_t c_{t+1} c_{t+2}$. Consequently, we have $b^d$ symbols since every $b$ symbol can be concatenated with every $b$ symbols including itself $d$ times. The difference from BOP is that the sliding windows are done at the beginning of the process to guarantee the online characteristic. After creating the new symbology sequence X, the probability $P \equiv p(b^d)$ of each symbol is calculated as

$$P = \frac{f_{b^d}}{W}, \tag{3.3}$$

Where $f_{b^d}$ are the symbols in $X$ and $W$ is the window length.

### 3.2.3 Jensen-Shannon distance calculation

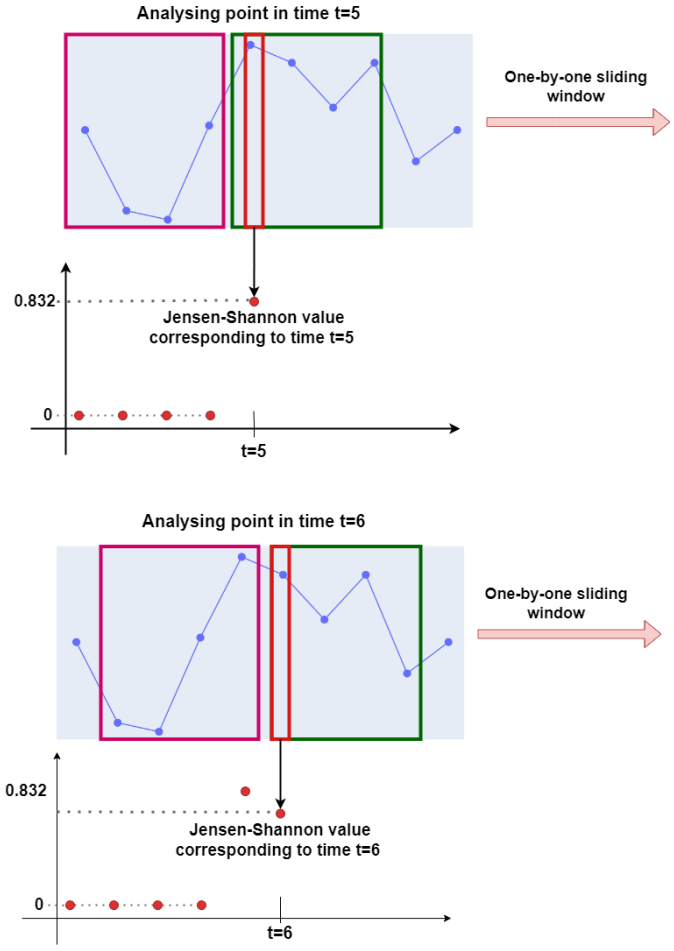
For each window $w_t$ and $w_{t+W}$, we calculate the Jensen-Shannon distance [44] between its probabilities distributions $\mathcal{P}_t$ and $\mathcal{P}_{t+W}$ following the equation:

$$\sqrt{\frac{D(\mathcal{P}_t||m) + D(\mathcal{P}_{t+W}||m)}{2}} \tag{3.4}$$

Where $D$ is the KL divergence described in Equation 2.1, and $m$ is the pointwise mean of $\mathcal{P}_t$ and $\mathcal{P}_{t+W}$. Jensen-Shannon is a distance measure. Thus, it has the advantage of being symmetric compared to dissimilarity measures [80] used in other CPD algorithms such as KL and PE divergence used in KLIEP, uLSIF, and RuLSIF, respectively.

We calculate the Jensen-Shannon distance measure in time stamp $t$ between the probabilities $\mathcal{P}_t$ and $\mathcal{P}_{t+W}$ of the two consecutive windows. Both consecutive windows move to the time stamp $t + 1$, and the Jensen-Shannon distance measure is calculated again.

Figure 3.6: Example of Jensen-Shannon construction in time $t = 5$ and $t = 6$.




Source: Created by the author.

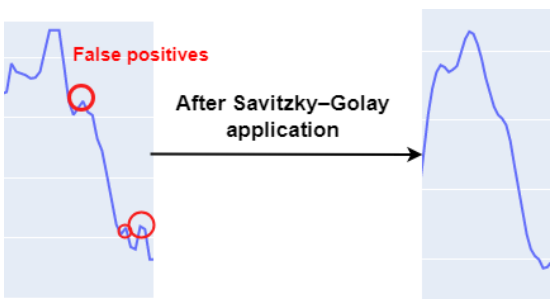
The Figure 3.6 illustrate how the Jensen-Shannon series are constructed based on the timestamps of the original time series how the windows slide through series generating the Jensen-Shannon information. First the two time series inside both windows (colored in pink and green) pass through the previous steps of SAX sequence (Section 3.2.1) and probability distribution calculation (Section 3.2.2). Then, the result of probability distribution is used in Jensen-Shannon calculation and it is added to the Jensen-Shannon results time series in the time $t$ corresponding to the time of first point (circulated in red) in the second time window.

### 3.2.4 Data smoothing and peak detection

After calculating the Jensen-Shannon distance, we apply the Savitzky–Golay filter. The Savitzky–Golay filter is a digital filter that can be applied to time series to smooth the series by fitting a low-degree polynomial with its adjacent data and using a linear least squares method [52]. The number of adjacency points $g$ is a parameter chosen by user definition, and the degree of polynomial is fixed at 3. To detect the peaks, we use the `argrelextrema` function in python [70], which calculates the relative extrema of data considering a neighborhood. For the size of the neighborhood, we set the value of $p$ for both sizes, so we have to look for $2p$ points ahead. Therefore, to evaluate if a point $t$ is a change point, our method requires $W + g + 2p$ points ahead.

The Savitzky–Golay filter is necessary to reduce the noise in order to be possible to use a smaller size of neighbors to detect peak without detecting False positive (small local peaks that does not represent a global peak). The Figure 3.7 shows an example of a small local peak circulated in Red and its disappearance after applying Savitzky–Golay filter.

Figure 3.7: Example of Savitzky-Golay filter reducing the number of False Positive peaks (red circles).




Source: Created by the author.

## 3.3 Complexity Analysis

In this Section, we detail the algorithm complexity analysis by scanning the pseudo-code in Algorithm 3.1. This algorithm provides a pseudo-code outline of our proposed framework SAXJS, including all the versions (SAXJS-S, SAXJS-T, SAXJS-$\tau$, and SAXJS-BOP). It takes as input a time series $\mathcal{S}(t)$, window-size $W$, SAX number of symbols $b$, number of jumps $\tau$, number of SAX symbols to aggregate $d$, number of neighbors $g$ and $p$, the name of the method chosen to calculate the probability distribution $c$ and the *threshold* to consider a CP or not. First, it is instantiated 3 lists. The first list, called "alljs" (Line 1), is created to keep all Jensen-Shannon (JS) results. The second "allsav" (Line 2) is created to save all JS results smoothed by the Savitzky-Golay filter. The last list (Line 3) is called "allresults" and keeps the final results whether it is a CP or not.

It then iterates the time series. The first operation inside the iteration in Line 4 is the SAX transformation executed in Line 5. This transformation is detailed in Section 3.2.1. As cited in [41], SAX operation has an $O(W)$ computational cost. After SAX transformation, in Lines 6 and 7, the SAX sequence is divided in two consecutive windows $sax1$ and $sax2$ of size $W$. Then, the probability distribution is calculated for both SAX sequences in Lines 8 and 9 using the Algorithm 3.2 detailed in Section 3.2.1.

As highlighted in Algorithm 3.2, there are four possible options to calculate probability distribution: SAXJS-S (from Line 2 to 8), SAXJS-T (from Line 10 to 16), SAXJS-$\tau$ (from Line 18 to 24) and SAXJS-BOP (from Line 26 to 32). All the options are divided into two steps: the first consists of counting the symbols' frequency in the SAX sequence, and the second consists of normalizing the vector or matrix of the symbol's frequency. As SAXJS-S iterates in all the points in the SAX sequence (Lines 3 and 4), its first step has the complexity of $O(W)$. For SAXJS-T (Lines 11 and 12) once we count the symbols connections, the last symbol would not have the next symbol to connect so that the complexity would be $O(W+1)$ For SAXJS-$\tau$ (Lines 19 and 20) and SAXJS-BOP (Lines 27 and 28), we iterate over $W-\tau$ and $W-(W \mod d)$ points, respectively. Thus, their complexity is $O(W-\tau)$ and $O(W-(W \mod d))$.

The second step of normalizing the frequency of each symbol depends on the symbology chosen in each version. For SAXJS-S (Lines 6 and 7), we count and normalize the frequency of each $b$ SAX symbol. Therefore, as we create an ordered dictionary and add 1 to the count of the symbols, it has the complexity of $O(b)$ to normalize all symbols. For SAXJS-T (Lines 14 and 15) and SAXJS-$\tau$ (Lines 16 and 17), we use the connection between 2 symbols as a symbology, so the number of possible connections is $b^2$ because each symbol can connect to any other symbol including itself. Being $b^2$ symbols, the complexity of the normalization is $O(b^2)$. In SAXJS-BOP (Lines 30 and 31), it is possible to choose the number of SAX symbols that will be concatenated to form one symbology

**Require:** Time series $\mathcal{S}(t) = \{s_t\}_{t=1}^{L}$, window-size $n$, SAX number of symbols $b$, number of jumps $\tau$, number of SAX symbols to aggregate $d$, number of neighbors $g$ and $p$, the method chosen to calculate the probability distribution $c$ and the *threshold* to consider a CP or not. Time series length $L$ must be greater than $n$.

$alljs \leftarrow []$
$allsav \leftarrow []$
$allresults \leftarrow []$
**for** $i \leftarrow 0$ to $(N - 2W)$ **do**
    saxseq $\leftarrow$ SAX sequence of size $2W$
    sax1 $\leftarrow$ sax1[i:$W$] ▷ sax1 is the sax sequence of the first window
    sax2 $\leftarrow$ sax1[i+$W$:i+$2W$] ▷ sax2 is the sax sequence of the second window
    $prob1 \leftarrow$ calculateProbabDistrib(sax1,$b$,$\tau$, $d$, $c$), algorithm 3.2 ▷ prob1 is the probability distribution of the first window
    prob2 $\leftarrow$ calculateProbabDistrib(sax2,$b$,$\tau$, $d$, $c$), algorithm 3.2 ▷ prob2 is the probability distribution of the second window
    jsresult $\leftarrow$ Jensen-Shannon(prob1,prob2)
    alljs.append(jsresult)
    **if** LENGTH(alljs) $\geq g$) **then**
        savresult $\leftarrow$ Savgovitz(allsav[:$g$])
        alljs.append(savresult)
    **end if**
    **if** LENGTH(allsav) $\geq 2p$ **then**
        resultsav $\leftarrow$ allsav[LENGTH(allsav) - $p$] ▷ Get the value in the last $p$ position of allsav
        **if** $resultsav \geq threshold$ **then**
            Add on allsavfilter only the values in the last $2p$ positions in allsav that are bigger than the threshold
            agrelresult $\leftarrow$ agrelextrema(allsavfilter[:$2p$])
            **if** resultsav == agrelresult **then**
                allresults.append(1) ▷ It is a CP
            **else**
                allresults.append(0) ▷ It is not a CP
            **end if**
        **else**
            allresults.append(0) ▷ The point is not a CP
        **end if**
    **end if**
**end forreturn** allresults

Algoritmo 3.1: SAXJS: proposed algorithm.

**Require:** A SAX sequence $\hat{C} = \{\hat{c}_1, \cdots, \hat{c}_W\}$, SAX number of symbols $b$, number of jumps $\tau$, number of SAX symbols to concatenate $d$ and the name of the method to use $c$

1: W ← LENGTH(saxseq)
2: **if** $c$ == "SAXJS-S" **then** ▷ Use the option described in Subsection 3.2.2.1
3: **for** i ← 0 to $W$ **do**
4: Add 1 to the SAX symbols that appear in position i of sax sequence saxseq.
5: **end for**
6: **for** i ← 0 to $b$ **do**
7: Normalize the SAX symbol in position i ▷ Equation 3.1
8: **end for**
**return** Probability distribution of each $b$ SAX symbols
9: **end if**
10: **if** $c$ == "SAXJS-T" **then** ▷ Subsection 3.2.2.2
11: **for** i ← 0 to $W - 1$ **do**
12: Add 1 to the connections between i and i+1 position in a graph of all possible SAX symbols connections.
13: **end for**
14: **for** i ← 0 to $b^2$ **do**
15: Normalize the SAX symbol connection in position i ▷ Equation 3.2
16: **end for**
**return** Probability distribution of each SAX symbols connection
17: **end if**
18: **if** $c$ == "SAXJS-$\tau$" **then** ▷ Subsection 3.2.2.2
19: **for** i ← 0 to $W - \tau$ **do**
20: Add 1 to the connections between i and i+1 position in a graph of all possible SAX symbols connections.
21: **end for**
22: **for** i ← 0 to $b^2$ **do**
23: Normalize the SAX symbol connection in position i ▷ Equation 3.2
24: **end for**
**return** Probability distribution of each SAX symbols connection
25: **end if**
26: **if** $c$ == "SAXJS-BOP" **then** ▷ Subsection 3.2.2.3
27: **for** i ← 0 to $W - (W \mod d)$ by $d$ **do**
28: Concatenate the SAX symbol in position i to its next $d$ symbols
29: **end for**
30: **for** i ← 0 to $b^d$ **do**
31: Normalize the SAX symbol concatenation in position i ▷ Equation 3.3
32: **end for**
**return** Probability distribution of each SAX symbols concatenation
33: **end if**

Algoritmo 3.2: Probability distribution calculation.

by a user-defined variable $d$. Hence, the complexity is $O(b^d)$. For the algorithm to work, $W$ must be much greater than the number of symbols ($b$ for SAXJS-S, $b^2$ for SAXJS-T and SAXJS-$\tau$ and $b^d$ for SAXJS-BOP) to have a sufficient frequency of each symbol and not to have a very sparse vector of probability distribution which would generate a not satisfactory Jensen-Shannon calculation. Therefore, the first step's complexity surpasses the second step's complexity in all the SAXJS options.

Then, the Algorithm 3.2 is finalized, and we come back to 3.1 Line 10 where the Jensen-Shannon calculation is executed. The Jensen-Shannon iterates over all the probability distribution vectors. Therefore, its complexity equals the number of symbols of each SAXJS variation. Its calculation is more detailed in Equation 3.4. The next part of the Savitzky-Golay filter application has a complexity of $O(g)$ corresponding to the number of neighbors $g$ as it fits a polynomial equation to smooth a point in time $t$ based on its neighbors. The filter is only applied when the minimal number of neighbors is achieved, as is seen in the condition in Line 21.

The peak detection step starts in Line 17 of Algorithm 3.1. It filters the allsav array by getting its last $2p$ values, which are bigger or equal to the *threshold*. As it gets the last $2p$ values to filter, this process has a $O(p)$ complexity. Then, the agrelextrema operation compares a point in time $t$ with some neighbors $p$ on both sides to analyze if it is the greater point and, then, if it can be a CP. Therefore, its complexity is $O(p)$.

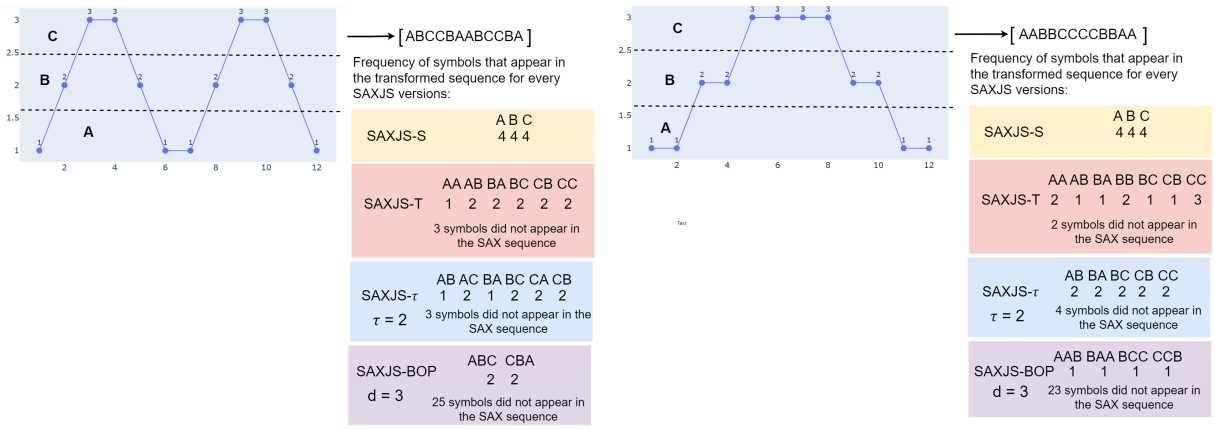
Summarizing all the operations inside the Line 4, we have the following complexity:

- SAX transformation (Line 5): $O(W)$
- Probability distribution (Lines 8 and 9):
  - SAXJS-S option: $O(W)$
  - SAXJS-T: $O(W-1)$
  - SAXJS-$\tau$: $O(W-\tau)$
  - SAXJS-BOP: $O(W-(W \mod d))$
- Jensen-Shannon calculation (Line 13): $O(g)$
- Peak detection (Line 17 to 21): $O(p)$

As the window size $W$ is always the largest user-defined value, the $O(W)$ complexity of SAX transformation dominates the other complexity of many symbols and some neighbors. The operation is done for every point of the time series $\mathcal{S}(t)$ of size $N$ except the first points in time where $N \geq 2W$ because the algorithm must have the two consecutive windows complete to work and be possible to compare its probability distributions. Nevertheless, the final complexity of the algorithm is $O(NW)$ as we run the complexity $O(W)$ $N$ times and $N >= 2W$.

## 3.4 Advantages and disadvantages

Figure 3.8: Example of two-time series with different frequencies and their respective SAX symbol sequences.




Source: Created by the author.

Figure 3.8 summarizes the difference between SAXJS versions and their disadvantages and advantages. Comparing the frequencies generated by each version from an SAX sequence, we can observe that SAXJS-S has the same distribution for both windows; therefore, it does not capture the difference of frequency shifts between the shape of the two-time series. On the other hand, SAXJS-T, SAXJS-$\tau$, and SAXJS-BOP can capture these changes, generating different distributions. However, in SAXJS-BOP, several symbols do not appear in the SAX sequence, which produces many zeros in the probability distribution, which can be a problem when measuring the distance between probability distributions.

In summary, our proposed online framework to detect CPs called SAXJS and its versions have the following characteristics:

- **Online**: all SAXJS variations are online, eliminating the necessity for the complete time series information and enabling it to run on streaming data.
- **It does not have a cross-validation step**: our framework does not have a step of cross-validation that may burden the process, such as RuLSIF.
- **Robustness to noise**: The application of time series discretization using SAX enhances algorithm robustness against noise by transforming noisy data into standardized symbols based on their distribution within the Gaussian curve.
- **Scalability**: Our framework is based on simple and fast transformation, with a $O(NW)$ computational cost. It allows its application in huge time series.

SAXJS and its versions also have some disadvantages, such as the following:

- **Design for only univariate data**: our framework is designed to deal with only 1-dimensional time series and does not work in multivariate data.
- **Needs more data in the sliding window**: To obtain a probability distribution with few zeros, the sliding window must be large enough so that most symbols appear in the SAX sequence. Consequently, the sliding window must have a length $W$ greater than other CPD algorithms require. As SAXJS-BOP requires more symbols, it needs more data in the sliding window than other methods.
- **The SAXJS-S captures worse frequency shift than the other versions**: As shown in Figure 3.8, the SAXJS-S does not capture frequency shifts satisfactorily compared to other SAXJS versions.

# Chapter 4

# SAXJS experiments Setup

In this chapter, we describe the quality metrics, the different data sets used in the experiments, and the parameter configurations used to evaluate the SAXJS compared to other state-of-the-art CPD algorithms. In Section 4.1, we explain the evaluation metrics. Section 4.3 describes all the different scenarios data sets, including synthetic data and real-world cases. Section 4.4 presents the hyperparameter configuration employed in the grid search for both SAXJS versions and other CPD methods.

## 4.1 Evaluation measures

This section explains the metrics used to evaluate the quality of the change point methods. We use the Receiver Operating Characteristic (ROC) curve and the area under the curve (AUC) (Section 4.1.1) and the detection delay (Section 4.1.2). In order to evaluate the cases without change points, we use a different way to calculate the False Positive Rate (FPR) as the one seen to create the ROC curve (Section 4.2).

### 4.1.1 Receiver Operating Characteristic (ROC) curve and the area under the curve (AUC)

Following the strategy used in [45], the Receiver Operating Characteristic (ROC) curve and the area under the curve (AUC) are calculated using the definition of true positive rate (TPR) and false positive rate (FPR) described in [30] as

- True positive rate (TPR): $\frac{n_{cc}}{n_{rc}}$
- False positive rate (FPR):$\frac{n_{al}-n_{cc}}{n_{al}}$

where $n_{cc}$ is the number of times the change point alarm was correct, $n_{rc}$ is the number of all ground truth change points, and $n_{al}$ is the number of all change points alarms.

A detection alarm at time $t$ is considered correct if a true change point exists at step $t*$ such that $[t*-I, t*+I]$, being $I$, the tolerance range. To avoid duplication, the $k$th alarm at the step $t_k$ if $t_k - t_{k-1} < 2I$ is removed.

Following the example in [45], the tolerance range $I$ for the synthetic data set with changes in every 100 steps is 10 for the synthetic data with changes in every 1000 steps $I = 100$. The HASC data set was tested with $I = 10$ and $I = 100$. For Bee dance and Well log, we use $I = 10$, and for Run-log, we use $I = 5$ following the example in [7].

### 4.1.2 Detection delay calculation

In addition to the ROC curve and the AUC value, we also evaluate the detection delay, which is the measure of how close each predicted change point is to the actual change point by the following equation 4.1 based on [3]. This metric helps measure the algorithm's accuracy in detecting the specific change point within the tolerance range. We only discuss detection delay when the interval tolerance is higher than 50 because very small tolerance intervals mean that the detection delay values for different algorithms do not have so many differences. However, the detection delays for data sets other than HASC with $I = 100$ and synthetic data with changes every 1000 steps can be seen in Appendix B.

$$delay = \frac{\sum_{i=1}^{n_{cc}} Predicted(CPs)_i - Actual(CPs)_i}{n_{cc}}. \tag{4.1}$$

## 4.2 False Positive Rate(FPR) for cases without changes

To assess scenarios devoid of change points, we employ an alternative method for computing the False Positive Rate (FPR) described in [3] with the Equation:

$$\frac{n_{FP}}{N},$$

where $n_{FP}$ is the total of false positives in our detection and $N$ is the time series size.

## 4.3 Data sets descriptions

This section comprehensively describes all the data sets employed in the experiments to evaluate the SAXJS algorithm, comparing it with other state-of-the-art methods. We leverage a total of 7 scenarios of data, as it aligns with the average number commonly utilized in the change point literature [14, 9, 35]. We choose a variety of data sets from various domain applications with different lengths, numbers of CPs, and CPs distances, as shown in Table 4.1 to demonstrate the adaptability of our algorithm across diverse scenarios. These 9 data are grouped into a total of 7 scenarios. The first group are synthetic data sets with changes every 100 points (JM 100, SV 100 and CC 100). The second are synthetic sata sets with changes every 1000 points(JM 1000, SV 1000, CC 1000). The third to sixth are the real world data: HASC Challenge 2011, Bee dance, Well Log and Run Log. Finally, the seventh group are the time series without change points with sizes of 5000 and 50000.

Table 4.1: Overview of data sets (synthetic and real-world cases). Mean and standard deviation are reported for data sets consisting of multiple time series.

| **Data set** | **Length** | **#series** | **#CPs** | **CPs distances** | | |
|---|---|---|---|---|---|---|
| | | | | **Q10** | **Q50** | **Q90** |
| No change 5000 | 5000 | 0 | 0 ± 0 | 0 | 0 | 0 |
| No change 50000 | 50000 | 0 | 0 ± 0 | 0 | 0 | o |
| JM 100 & SV 100 | 4998 | 2 | 49 ± 0 | 100 | 100 | 100 |
| CC 100 | 4862 | 1 | 48 | 100 | 100 | 100 |
| JM 1000 & SV 1000 | 49998 | 2 | 49 ± 0 | 1000 | 1000 | 1000 |
| CC 1000 | 48968 | 1 | 48 | 1000 | 1000 | 1000 |
| Bee dance | 827 ± 202 | 6 | 20 ± 4 | 19 | 45 | 77 |
| HASC-2011 | 10931 ± 229 | 5 | 12 ± 0.4 | 755 | 917 | 1029 |
| Well log | 4050 | 1 | 11 | 59 | 180 | 515 |
| Run Log | 376 | 1 | 8 | 18 | 36 | 59 |

In Section 4.3.1, we explain the creation of the synthetic data sets used to evaluate SAXJS performance in a controlled environment. The purpose of assessing the algorithm in a controlled environment is to ascertain its capability to detect fundamental changes in time series such as mean, variance, and frequency, while also ensuring it refrains from detecting changes in cases where the series remains without change points. After evaluating the quality of SAXJS in a controlled environment, it is crucial to assess its performance in

real-world scenarios to determine the practical viability of our algorithm. We describe the data sets from real-world cases in Sections 4.3.2.

### 4.3.1 Synthetic data sets

To evaluate SAXJS in a controlled environment, we created eight different time series of synthetic data. In the first and second, we create time series with size 5000 and 50000, respectively, free of changes. Then, in the third and fourth time series, we inserted mean changes every 100 steps and in the second time series, we inserted mean changes in every 1000 steps. The fifth and sixth series exhibited scaling variance alterations at intervals of 100 and 1000 steps, and in the seventh and eighth, we inserted coefficient changes in every 100 and 1000 steps. The following is the description of the synthetic bases for every change category analyzed as suggested in [45, 14]:

- **No change**: The following 1-dimensional model borrowed from [74] is used to generate approximately 5000 samples (i.e., t = 1, . . . , 5000) and approximately 50000 samples (i.e., t = 1, . . . , 50000):

$$y(t) = 0.6y(t-1) - 0.5y(t-2) + \epsilon_t,$$

  where $\epsilon_t$ is a Gaussian noise with mean $\mu$ and standard deviation equal to 1.5. The initial values are set as $y(1) = y(2) = 0$. As the series does not have change points, $\mu$ and standard deviation never change.

- **Jumping mean (JM)**: The same model of the data set with no changes is used with mean $\mu$ and standard deviation 1.5. A change point is inserted every 100-time steps or 1000-time steps increasing the intensity of the jumping mean change by modifying $\mu$ at time $t$ according to

$$\mu_N = \begin{cases} 0, & N = 1, \\ \mu_{N-1} + \frac{N}{16}, & N = \{2, ..., 49\}, \end{cases}$$

  where $N$ is a natural number such that $100(N-1) + 1 <= t <= 100N$.

- **Scaling variance (SV)**: The same model is used, but a change point is inserted every 100 or 1000 steps increasing the intensity of the variance change by modifying the standard deviation $\alpha$ at time $t$ as

$$\alpha = \begin{cases} 1, & N = \{1, 3, ..., 49\} \\ \ln(\epsilon + \frac{N}{4}), & N = \{2, 4, ..., 48\} \end{cases}.$$

- **Changing coefficient (CC)**: The model used to generate the series follows the equation described in [14]:

$$y(t) = y(t-1) + \epsilon_t,$$

where $\epsilon_t$ Gaussian noise with standard deviation set to 1.5 and $mu$ set to 0. For every segment in 100 or 1000 steps, the coefficient $y$ is alternatively sampled from $U([0, 0, 5])$ and $U([0.8, 0.95])$ to generate clear differences in autocorrelation and frequency content between segments. The objective of this synthetic dataset is to evaluate our algorithm in frequency changes.

### 4.3.2 Real-world cases

As shown in Figure 4.1, we evaluate SAXJS using four real-world cases. The first real-world case is a change point benchmark data set [2] **HASC Challenge 2011** [28] also used to evaluate TIRE [14] and RuLSIF [45]. HASC Challenge 2011 is a collaborative project to collect human activity data from accelerometer sensors on wearable devices. Each participant in the project should gather at least 5 out of the 6 activities for at least 5 seconds: stay, walk, jogging, skip, stair-up, and stair-down.

The total time of the activities should be at least 120 seconds. The challenge has a committee to evaluate whether the annotation is correct. The HASC Challenge 2011 [29] gathered and composed 4897 accelerometer data with 116 subjects. As the accelerometer data is 3-dimensional, we use the l2-norm transformation following [45]. We evaluate the 5 first participants' accelerometer data, as outlined in the reference [45].

The second real-world case is the **Bee dance** [50] data set, which consists of six 3-dimensional time series of the bee positions (location in 2D plane and angle differences). As the data are 3-dimensional, we use the l2-norm transformation as well. Information on the bee's position makes it possible to identify the bee's dance, which is classified as left turn, right turn, and waggle. The identification of bees' dance is of interest to ethnologists as its length (duration) and representation correspond to the distance and the orientation of the food source. Bee dance is often used to evaluate CPD algorithm [9, 11, 68, 69, 72] and TIRE [14].

The last real-world scenarios are **Well log** and **Run Log**. They are 2-time series part of the Alan Turing Institute project [69] to create a data set with benchmarks specifically tailored for the evaluation of CPD algorithms. The Well log data set, initially examined in [56], comprises nuclear magnetic resonance measurements acquired during

the drilling of a well. Changes in the time series' mean correlate with alterations in rock stratification. This benchmark was also used in the TIRE evaluation. The Run Log data set represents a runner's pace during an interval training session with instructions alternating between run and walk. This data was also used to evaluate ADAGA [7].

## 4.4 Parameters settings

This section presents an overview of the hyperparameters utilized in the grid search for each SAXJS version and state-of-the-art CPD method across various scenarios, including synthetic data and real-world cases. Each CPD algorithm requires the configuration of distinct hyperparameters with different meanings. In the Sections 4.4.1 to 4.4.5, we describe the hyperparameter configuration of every CPD algorithm for all scenarios. Following is the list of all CPD methods with their respective hyperparameter meanings:

- SAXJS:
  - $W$: window size of the SAX transformation.
  - $b$: number of SAX symbols.
  - $g$: number of neighbors to fit the low-degree polynomial in the Savitzky–Golay filter application.
  - $p$: number of neighbors to compare in the peak detection.
  - Threshold: threshold for defining CPs.
  - $d$: number of SAX symbols to aggregate in one symbol (only for the SAXJS-BOP version).
  - $\tau$: number of jumps to establish the connection between symbols (only for the SAXJS-$\tau$ version).
- KernelCPD:
  - Kernel: the kernel model.
  - Jump: controls the grid of possible change points.
  - min-size: minimal segment length to prune the data set.
  - pen: The $penalty > 0$ to be used in the optimization function. The lower the pen, the more CPs are found.
- RuLSIF:

  - $K$: number of kernels.
  - $\sigma$: controls the relative PE-divergence. The lower the $\sigma$, the more the density ratio is plain and less "smooth".
  - $W$: window size of sequential windows to be compared.
  - $g$: We use the same Savitzky–Golay filter application as it is used in SAXJS.
  - $p$: We use the same peak detection as in SAXJS.
  - Threshold: threshold to define CPs.

- TIRE
  - $W$: window size of sequential windows to be compared.
  - $M$: length of Fourier transform.
  - Invar: number of time-invariant features.
  - $h$: number of features ($timeinvariant + instantaneous$).
  - $Ka$: number of Autoencoders.
  - epoch: number of epochs to train the neural network.
  - $\delta$: if its to use time domain.
  - $\beta$: if its to use frequency domain.
  - Threshold: threshold to define CPs.

  TIRE creators use two versions of these hyperparameter values: *v1* and *v2*. These two versions are tested with only time domain features, frequency domain features, and both.

- BOCPDMS
  - Intensity: intensity of the Hazard function.
  - *Gamma* $\alpha_0$: $\alpha$ parameter of the Inverse Gamma.
  - *Gamma* $\beta_0$: $\beta$ parameter of the Inverse Gamma.
  - Threshold: threshold to define CPs.

### 4.4.1 Synthetic cases (interval 100)

The first scenario encompasses synthetic data with mean, variance, and frequency changes occurring regularly at intervals of either 100 or 1000 steps. The versions with

changes every 100 steps and every 1000 steps have unique hyperparameter configurations. In this subsection, we focus on the version with change points at regular intervals of 100 steps. For the change in every 100 steps, we use the following parameter settings for SAXJS versions and other CPD algorithms:

- **SAXJS**: For every SAXJS version, we use the same $W$: (50,60,70,80,90,100), $g$: (11,21,31,41), $p$: (5,10,15) and Threshold from 0.3 to 0.8 by 0.01. Compared to the configuration used in every 1000 steps case, the values of $W$, $g$, and $p$ change due to the smaller length and CPs distance of synthetic data sets with changes every 100 points. For SAXJS-S, SAXJS-T and SAXJS-$\tau$, we use $b$: (3,4,5,6,7,8,9). For the SAXJS-BOP, we use $b$: (3,4,5,6,7) due to the memory increase described in the Chapter 2 by using the $d$: (2,3). For the SAXJS-$\tau$, we use 2 to 10 by 1.

- **KernelCPD**: We tested the three available Kernel options: (linear, cosine, Gaussian). We use the values from 1 to 10 by 1 for the Jump. The min-size varied from 10 to 50 by 10. The pen values varied from 1 to 55 by 5.

- **RuLSIF**: We used the same parameter settings of $\sigma : (0.1, 0.2)$ and $k : (5, 10, 15)$ as suggested in [45]. The window size $W$ was set according to the length of the data set and the CPs distance. Therefore, for the change in every 100 steps, $W$ varies from 50 to 100 by 10. To run the method online, we applied the same data smoothing and peak detection strategy as SAXJS and used the same hyperparameters configurations of $g$: (11,21,31,41), $p$: (5,10,15). As the PE divergence results scale differs from the Jensen-Shannon distance, the Threshold varies from 0.1 to 2 by 0.1.

- **TIRE**: For all the scenarios, we use the same configuration set described in [14] as the majority of the data sets largely overlap with ours, and the sensitivity analyses of TIRE shows that its good performance does not critically depends on its hyperparameters. The only hyperparameter we changed is the threshold ranging from 0.05 to 0.55 by 0.01.

- **BOCPDMS**: For all the scenarios, we use the suggestion of hyperparameters configuration described in [69]:

  - Intensity: (10, 50, 100, 200)
  - *Gamma* $\alpha_0$: (0.01, 0.1, 1, 10, 100)
  - *Gamma* $\beta_0$: (0.01, 0.1, 1, 10, 100)

  Gerrit JJ Van den Burg and Christopher KI Williams [69] used 30 minutes as the algorithm time out, but because our time series has a larger length, we used 3 hours as the time out. The threshold change is different for each case. The threshold for the synthetic data set with changes every 100 intervals was from 50 to 400 by

50. These values were determined through an empirical test, which compared the threshold options from 1 to 10 with a range of 50 to 400.

### 4.4.2 Synthetic cases (interval 1000) and real-world case: HASC Challenge 2011

As the CPs distances of the synthetic cases with changes every 1000 steps and HASC Challenge 2011 are comparable, we utilize the same following parameter settings:

- **SAXJS** For every SAXJS version, we use the same $W$ values from 300 to 800 by 100, $g$: (81,101,201), $p$: (5,10,15,20), and Threshold from 0.3 to 0.8 by 0.01. Compared to the configuration used in every 100 steps case, the values of $W$, $g$, $p$, and $\tau$ change due to the larger length and CPs distance of synthetic data sets with changes every 1000 points. The $\tau$ varies from 2 to 15 by 1. We use the same configuration set for the hyperparameters $b$, $d$, and Threshold as the other scenarios.
- **KernelCPD**: The Kernel, Jump, and pen values are the same for every data set. The min-size varied from 50 to 150 by 10.
- **RuLSIF**: We use the same hyperparameter configuration described in [45], except for $W$. The window size $W$, $g$, and $p$ are always the same as SAXJS.
- **TIRE**: We use the same *v1* and *v2* configuration as the one described in [14] and vary the threshold from 0.05 to 0.55 by 0.01.
- **BOCPDMS**: For all the scenarios, we use the suggestion of hyperparameters configuration described in [69], except for the threshold. For this case, we use the threshold of 50 to 400 by 50 for HASC and 1 to 10 for synthetic data.

### 4.4.3 Real-world cases: Bee dance and Run log

For all the SAXJS and other CPD algorithms, we utilize the same hyperparameter configuration for the Bee dance and the Run log real-world cases, as both data sets are comparable in CPs distance. The only hyperparameters that can change are the ones related to window sizes, as the Bee dance time series is longer than Run-log time series.

- **SAXJS** For every SAXJS version, we use the same $W$: (30,40,50,60,70,80,90,100), $g$: (5,11,15), $p$: (5,10,15) due to the length and CPs distance of the series. We use the same configuration set for the hyperparameters $b$, $d$, and Threshold as the other scenarios. For the $\tau$, we use 2 to 10 by 1 due to the smaller length of the series in comparison to HASC and synthetic data with changes in every 1000 steps length.

- **KernelCPD**: The Kernel, Jump, and pen values are the same for every data set. For the Run-log, the min-size varied from 2 to 90 by 2. For the Bee dance, the min-size varied from 10 to 140 by 10, including 5. For the Run log, the min-size varied from 2 to 30 by 2 and from 30 to 90 by 10.

- **TIRE**: We use the same *v1* and *v2* configuration as the one described in [14] and varied the threshold from 0.05 to 0.55 by 0.01. As the Run-log is the only data set not used to evaluate TIRE, we test the window-size: (20, 30, 40, 50, 60, 70, 80, 90, 100).

- **RuLSIF**: We use the same hyperparameter configuration described in [45], except for $W$. The window size $W$, $g$ and $p$ are always the same as SAXJS.

- **BOCPDMS**: As Bee Dance is also used in the BOCPDMS evaluation, we used the same parameter configuration described in [35]. The threshold was the only parameter different. We used thresholds: (50, 100, 150, 200, 250, 300). We use the threshold from 1 to 10 by 1 for the Run log.

### 4.4.4 Real-world cases: Well log

The last real-world case to be explored is the Well log data set. For this data set, we use the following parameter configuration for every CPD algorithm:

- **SAXJS**: For every SAXJS version, we use the same $W$: (60, 70, 80, 90, 100, 200, 300, 400), $g$: (5, 15, 25, 85), $p$: (5, 10, 15, 20). We use the same configuration set for the hyperparameters $b$, $d$, and Threshold as the other scenarios. For the $\tau$, we use 2 to 10 by 1.

- **KernelCPD**: The Kernel, Jump, and pen values are the same for every data set. The min-size varied from 10 to 140 by 10, including 5.

- **TIRE**: We use the same *v1* and *v2* configuration as the one described in [14] and varied the threshold from 0.05 to 0.55 by 0.01.

- **RuLSIF**: We use the same hyperparameter configuration described in [45], except for $W$. The window size $W$, $g$ and $p$ are always the same as SAXJS.
- **BOCPDMS**: For all the scenarios, we use the suggestion of hyperparameters configuration described in [69], except for the threshold. We use the threshold from 1 to 10 by 1 for this case.

### 4.4.5 Synthetic cases free of changes

For this data set, we employed a distinct approach aimed at assessing our Framework's ability to sustain robustness and refrain from detecting change points in series where they are absent. As such, we exclusively evaluate our Framework using the optimal values identified in each Grid Search with the minimal Threshold of 0.30 that we use in all the experiments. Larger data sets, such as Synthetic data (changing every 1000 points), HASC Challenge 1000 are tested with the time series free of changes of length of 50000. Conversely, for smaller data sets such as Synthetic data (changing every 100 points), Bee Dance, Well Log and Run Log we utilize the series with no changes of length 5000.

# Chapter 5

# SAXJS evaluation with different scenarios

Our main goal, as described in Section 1.2, is to propose a novel CPD framework that is online, and adaptable to different systems. To investigate if we achieve this goal, in this Chapter, we evaluate our proposed framework performance and adaptability in different scenarios using the data sets and metrics detailed in Chapter 4. By evaluating our algorithm in different scenarios we are capable of answering the following three questions:

- Is SAX symbols' distribution capable of providing good change point detection?
- Is SAX symbols' transition distribution capable of providing good change point detection?
- Is Bag-of-Patterns (BOP) based distribution capable of providing good change point detection?

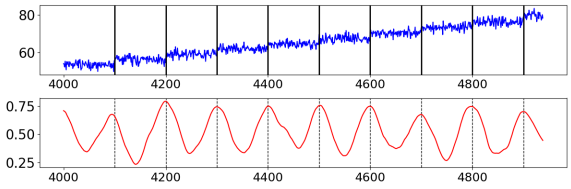
We believe that the SAX symbols', SAX symbols' transitions, and BOP distributions, based on their characteristics (discussed in Chapter 3), are transformations capable of highligh a time series's underlying patterns that lead to CPs. This section aims to evaluate the capability of these three models in detecting CPs in synthetic and real-world scenarios by comparing its results with state-of-the-art methods. To fine tune the algorithms, we adopted a Grid Search approach on the hyperparameters described in Chapter 4 and compare the best models of each state-of-art methods.

In Section 5.1, we investigate the capability of our framework to detect CPs in a synthetic data set that includes changes in mean, variance, and coefficient. In Section 5.2, 5.3, 5.4 and 5.5, we test our algorithm with real-world CPs cases of accelerometer sensors, bee dance sensors, nuclear magnetic resonance and pace of a runner during a training session, respectively. In Section 5.6, we tested our algorithm capability of not detecting false positives in a series with no change points. In Section 5.7, we present the Final Remarks of the chapter by comparing and discussing the results of our framework and its variations in the different scenarios.

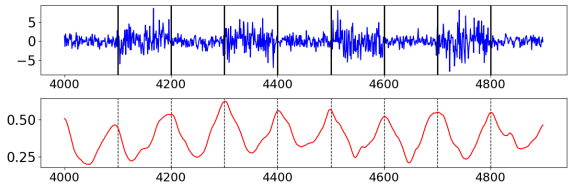
## 5.1 Synthetic cases: Jumping mean, Scaling variance, and Changing coefficient

First, we evaluate our framework using six synthetic data sets with manually inserted change points every 100 and 1000 steps to analyze the SAXJS capability to detect mean, variance, and coefficient changes (also known as frequency changes).

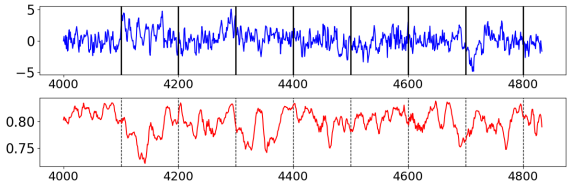
Figure 5.1: (Synthetic data with change in every 100 steps). Illustrative time series samples (upper) and the change point score obtained by SAXJS-T (lower). Black vertical lines mark the true change points.




(a) JM 100




(b) SV 100



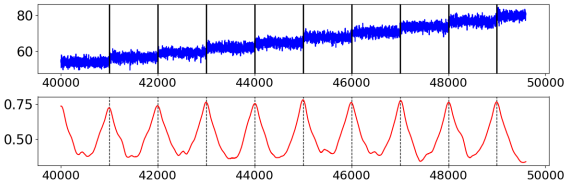

(c) CC 100

Source: Created by the author.

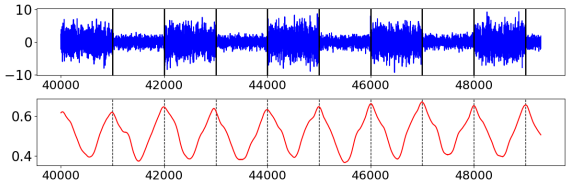
In Figures 5.1a, 5.1b, and 5.1c, we plotted samples of the synthetic data sets with changes in every 100 steps (blue curves), true change points (black vertical lines), and the change point score (red curves) obtained by SAXJS-T, the method with the biggest average AUC. Analyzing the plots, we can see that the algorithm captures the mean and variance changes, while there are more false positives for the change coefficient data set, as shown in Figure 5.1c. In Figures 5.2a, 5.2b, and 5.2c, we can see that the algorithm seems to capture all the changes, including the changing coefficient better, having the peaks synchronized with the real change points.

Observing Table 5.1 and 5.2, we can see the AUC values for the synthetic data set with changes in every 100 steps and every 1000 steps, respectively. The results show that the SAXJS variations perform better in scaling variance and changing coefficients. The jumping mean data set also has a satisfactory performance with an AUC higher than 0.8 for all SAXJS versions, considering changes every 100 and 1000 steps. However, it was inferior to TIRE results.

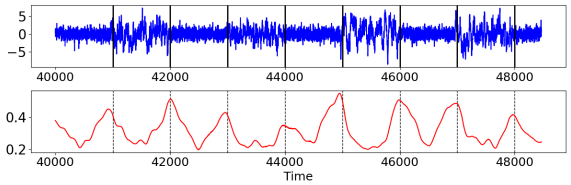
Figure 5.2: (Synthetic data with change in every 1000 steps). Illustrative time series samples (upper) and the change point score obtained by SAXJS-T (lower). Black vertical lines mark the true change-points.




(a) JM 1000




(b) SV 1000




(c) CC 1000

Source: Created by the author.

Table 5.1: AUC values for synthetic datasets with changes every 100 steps

| | JM 100 | SV 100 | CC 100 | Average AUC |
|---|---|---|---|---|
| KernelCPD | 0.830 | 0.307 | 0.313 | 0.483 |
| RuLSIF | 0.862 | 0.730 | **0.413** | 0.668 |
| TIRE | **0.906** | **0.881** | 0.385 | **0.724** |
| BOCPDMS | 0.260 | 0.818 | 0.290 | 0.456 |
| SAXJS-S | 0.850 | 0.807 | 0.436 | 0.698 |
| SAXJS-T | 0.882 | 0.922 | 0.498 | **0.767** |
| SAXJS-$\tau$ | **0.892** | **0.940** | 0.324 | 0.719 |
| SAXJS-BOP | 0.835 | 0.878 | **0.537** | 0.75 |

When evaluating a smaller data set with the changes happening every 100 points, all the competitors' average AUC, except for TIRE, were below 0.7, and all SAXJS variations' AUC were larger than 0.7, except for SAXJS-S. The highest average AUC value was the SAXJS-T of 0.767, followed by SAXJS-BOP with an AUC of 0.75 and TIRE with an AUC of 0.724.

For the data set with changes in every 1000 steps, the SAXJS variation AUC average was better than its competitors, except for RuLSIF in comparison to SAXJS-S. RulSIF showed satisfactory results with an average AUC of 0.905 which is larger than SAXJS-S of 0.793. The best result was the SAXJS-BOP with an average AUC of 0.968, followed by an average AUC of 0.950 for the SAXJS-T method. As expected, the SAXJS-BOP has a better result when the data set is larger because the aggregation of SAX symbols generated more symbols, requiring a larger data set to have a considerable frequency of every symbol. Analyzing Table 5.3, we can observe that SAXJS variations have a smaller

Table 5.2: AUC values for synthetic datasets with changes every 1000 steps

| | JM 1000 | SV 1000 | CC 1000 | Average AUC |
|---|---|---|---|---|
| KernelCPD | 0.953 | 0.123 | 0.579 | 0.552 |
| RuLSIF | 0.936 | **0.949** | 0.830 | **0.905** |
| TIRE | **0.980** | 0.756 | 0.587 | 0.774 |
| BOCPDMS | 0.326 | 0.948 | **0.978** | 0.751 |
| SAXJS-S | 0.816 | 0.796 | 0.766 | 0.793 |
| SAXJS-T | 0.898 | 0.945 | 0.970 | 0.938 |
| SAXJS-$\tau$ | 0.945 | 0.954 | 0.951 | 0.950 |
| SAXJS-BOP | **0.949** | **0.978** | **0.976** | **0.968** |

Table 5.3: (Synthetic data with changes in every 1000 points). Detection delay comparison of each CPD method with best parameters setting according to AUC.

| | JM 1000 | SV 1000 | CC 1000 | Average Delay |
|---|---|---|---|---|
| KernelCPD | **8.09** | 62.10 | 43.21 | 37.80 |
| RulSIF | 9.67 | **11.16** | 39.20 | **20.01** |
| TIRE | 15.81 | 76.59 | 53.19 | 48.53 |
| BOCPDMS | 22.75 | 18.84 | **35.34** | 25.64 |
| SAXJS-S | **6.39** | **8.86** | 33.20 | **16.15** |
| SAXJS-T | 7.69 | 13.27 | 29.39 | 16.78 |
| SAXJS-tau | 8.84 | 11.79 | 29.93 | 16.85 |
| SAXJS-BOP | 16.55 | 16.35 | **29.04** | 20.65 |

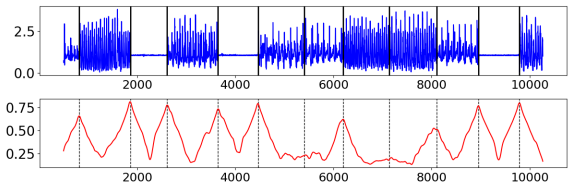
average delay than its competitors, except for SAXJS-BOP that has a larger delay than RulSIF.

## 5.2 Real-world cases: HASC Challenge 2011

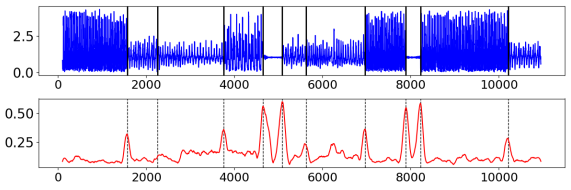
We tested our framework in four different real-world databases. In this Section, we discuss about the tests in the first real-world database. The first is the HASC (Human Activity Sensing Consortium) Challenge, a collaborative project to collect accelerometer sensor data in wearable devices. The accelerometer data have 3-dimensional information that we use l2-norm, which is the square root of the sum of the squared vectors, to transform in 1-dimensional following [45]. We evaluate our method with the five first different participant information: person101, person102, person103, person104, and person105, as outlined in the reference [45].

In Figures 5.3a, 5.3b, 5.3c, 5.3d and 5.4, we plotted examples of HASC original time series (blue curve), true change points (vertical black lines), and the change point score obtained by SAXJS (red curves). Observing the Figures 5.3a, 5.3c, 5.3d and 5.4, it

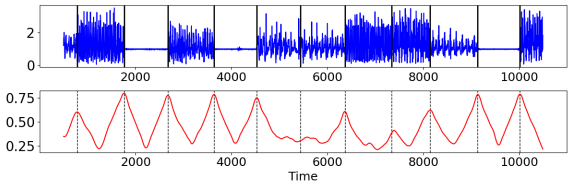
Figure 5.3: (HASC Challenge data from person101 to person104) Illustrative time series samples (upper) and the best change point score obtained by our proposed method (lower). Black vertical lines mark the true change points.



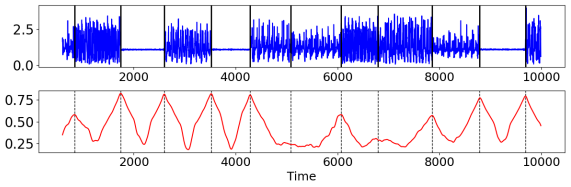

(a) Person101




(b) Person102



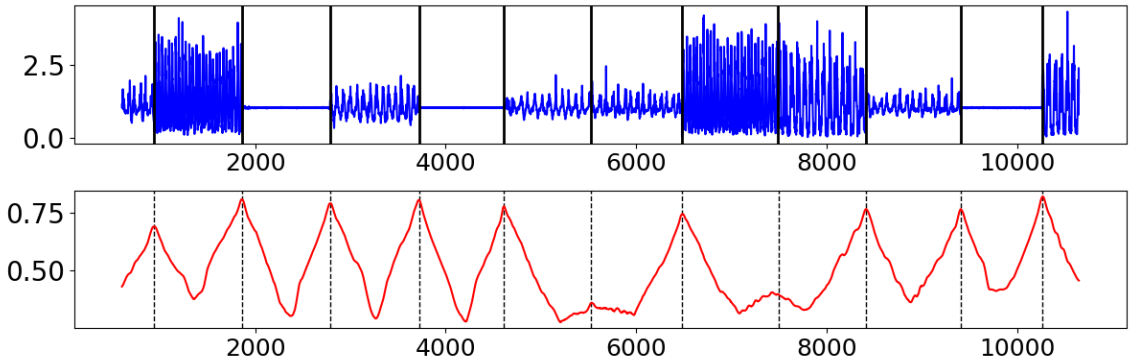

(c) Person103




(d) Person104

Source: Created by the author.

Figure 5.4: (HASC Challenge data person105). Illustrative time series samples (upper) and the best change point score obtained by our proposed method (lower). Black vertical lines mark the true change-points.




Source: Created by the author.

is possible to conclude that Person101, Person103, Person104 and Person105 have very similar behaviours. It shows that the algorithm captures all changing behaviors but has difficulty capturing the frequency changes between 5000 and 6000 and 7000 and 8000, which is difficult to visualize even for human eyes. Comparing these four results, Person103 has the most prominent Jensen-Shannon distance (red curve) peak between 7000 and 8000. The Person102 (Figure 5.3b) is the most different participant compared to others, and it has more variant CPs distances. Analyzing its Figure, we can see false positive CPs between timestamps 6000 and 7000 and the difficulty in capturing the variance change between timestamps 2000 and 3000.

Analyzing the results in Table 5.4 considering the tolerance interval of $I = 100$,

Table 5.4: AUC values for HASC Challenge using $I = 10$.

| | person101 | person102 | person103 | person104 | person105 | Average |
|---|---|---|---|---|---|---|
| KernelCPD | **0.664** | 0.545 | **0.710** | 0.605 | 0.544 | **0.619** |
| RuLSIF | 0.557 | 0.404 | 0.633 | 0.485 | 0.558 | 0.527 |
| TIRE | 0.074 | 0.008 | 0.536 | 0.010 | 0.109 | 0.147 |
| BOCPDMS | 0.523 | **0.571** | 0.545 | **0.667** | **0.597** | 0.581 |
| SAXJS-S | 0.778 | **0.818** | 0.759 | 0.774 | 0.875 | 0.801 |
| SAXJS-T | 0.804 | 0.795 | 0.811 | 0.788 | 0.875 | 0.815 |
| SAXJS-$\tau$ | **0.845** | 0.801 | **0.837** | **0.863** | 0.875 | **0.844** |
| SAXJS-BOP | 0.829 | 0.732 | 0.805 | 0.755 | **0.879** | 0.8 |

Table 5.5: AUC values for HASC Challenge using $I = 100$.

| | person101 | person102 | person103 | person104 | person105 | Average |
|---|---|---|---|---|---|---|
| KernelCPD | **0.978** | **0.954** | **0.997** | **0.967** | **0.929** | **0.965** |
| RuLSIF | 0.875 | 0.946 | 0.864 | 0.875 | 0.875 | 0.888 |
| TIRE | 0.863 | 0.848 | 0.917 | 0.917 | 0.917 | 0.892 |
| BOCPDMS | 0.799 | 0.812 | 0.839 | 0.872 | 0.782 | 0.821 |
| SAXJS-S | 0.875 | 0.864 | 0.882 | 0.875 | 0.875 | 0.874 |
| SAXJS-T | 0.875 | 0.930 | 0.917 | 0.899 | 0.875 | 0.899 |
| SAXJS-$\tau$ | 0.917 | **0.955** | **0.958** | **0.917** | 0.917 | **0.933** |
| SAXJS-BOP | **0.929** | 0.880 | 0.942 | 0.902 | **0.949** | 0.920 |

Table 5.6: (HASC Challenge 2011 data). Detection delay comparison of each CPD method with best parameters setting according to AUC for tolerance interval $I = 100$.

| | person101 | person102 | person103 | person104 | person105 | Average |
|---|---|---|---|---|---|---|
| KernelCPD | 43.25 | 16.82 | 12.42 | 47.75 | 51.58 | 34.36 |
| RulSIF | **15.33** | 35.20 | 16.00 | 15.44 | 14.56 | 19.31 |
| TIRE | 38.56 | 33.12 | 22.60 | 31.80 | 33.12 | 31.84 |
| BOCPDMS | 23.55 | **14.44** | **10.78** | **5.5** | **14.22** | **13.70** |
| SAXJS-S | 22.56 | **8.75** | 13.30 | 20.78 | 12.89 | 15.66 |
| SAXJS-T | 25.33 | 12.00 | 15.00 | 19.60 | **9.22** | 16.23 |
| SAXJS-tau | **9.27** | 21.30 | 14.18 | **8.70** | 13.10 | **13.31** |
| SAXJS-BOP | 14.64 | 16.50 | **13.27** | 20.73 | 11.45 | 15.32 |

it is possible to conclude that all SAXJS variations have competitive results. However, in most cases, it is not better than KernelCPD, and SAXJS-S is worse than TIRE and RuLSIF. But considering a more restrictive interval such as $I = 10$ showed in Table 5.5, the SAXJS has better results than other methods due to its low delay to find the change point as shown in the delay time mean in Table 5.6. All SAXJS versions have a detection delay time smaller than the other algorithms, except for the BOCPDMS. Despite the slight detection delay of BOCPDMS, it did not perform satisfactorily in CPs detection, considering a tolerance interval of $I = 10$.

SAXJS-$\tau$ achieves the best result of all SAXJS options in both interval tolerance. Despite its superior performance, all versions exhibit similar average AUC results, with

the largest difference of AUC average being 0.059 observed between the average AUC of SAXJS-S and SAXJS-$\tau$ within the tolerance interval $I = 100$. Furthermore, across all tolerance intervals, the average AUC values for all versions consistently exceed 0.8, indicating a satisfactory outcome.

## 5.3 Real-world cases: Bee dance

In this Section, we present and discuss about the results of the second real-world database. The second real-world data set was a set of bee positions used to identify the changes in bee dance. The Figures 5.5a, 5.5b, 5.5c, 5.5d, 5.5e and 5.5f shows the 6 sequences of bee positions we use to test our framework. Analyzing its Figures, we notice that the changes are more subtle and smooth than the other CPs data set we analyzed. The subtlety of changes can be the reason why the AUC results for SAXJS variations and other CPD methods are not as satisfactory as for other datasets, especially for Sequence 1, 2, and 3. For these three Sequences, the only method with good results with AUC larger than 0.8 was RuLSIF, and SAXJS-$\tau$ was the only one besides RuLSIF with an AUC larger than 0.75.

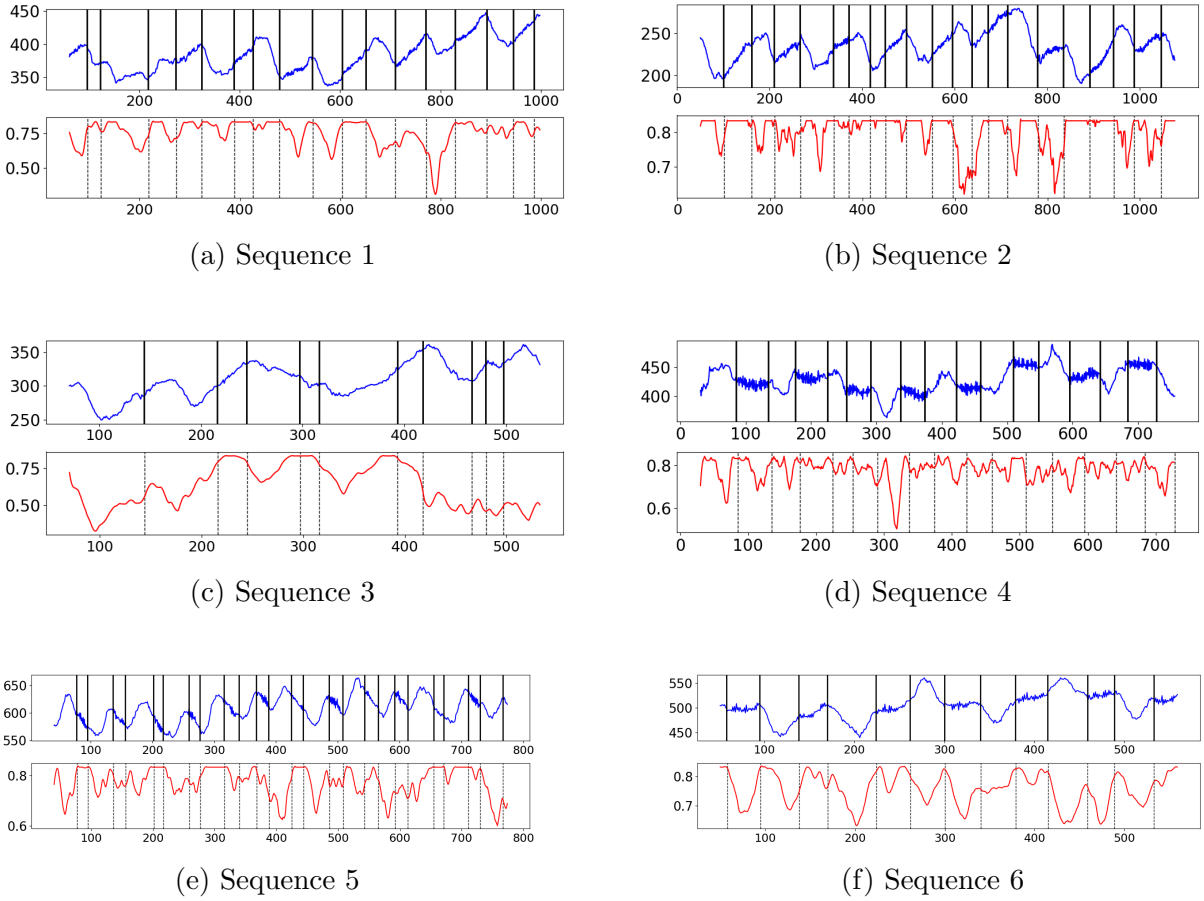
Table 5.7: AUC values for Bee Dance using $I = 10$.

| | Seq1 | Seq2 | Seq3 | Seq4 | Seq5 | Seq6 | Average |
|---|---|---|---|---|---|---|---|
| KernelCPD | **0.621** | **0.634** | 0.678 | 0.660 | 0.750 | 0.660 | 0.667 |
| RuLSIF | 0.588 | 0.619 | **0.808** | **0.730** | **0.789** | **0.857** | **0.732** |
| TIRE | 0.323 | 0.365 | 0.225 | 0.436 | 0.340 | 0.663 | 0.392 |
| BOCPDMS | 0.344 | 0.281 | 0.394 | 0.254 | 0.537 | 0.584 | 0.399 |
| SAXJS-S | 0.497 | 0.472 | 0.681 | 0.690 | 0.509 | 0.955 | 0.634 |
| SAXJS-T | 0.521 | 0.521 | 0.692 | 0.730 | 0.727 | 0.953 | 0.691 |
| SAXJS-$\tau$ | **0.698** | **0.683** | **0.782** | 0.750 | **0.883** | **1.0** | **0.799** |
| SAXJS-BOP | 0.561 | 0.637 | 0.719 | **0.782** | 0.811 | 0.855 | 0.728 |

The poor performance of the SAXJS variations in these 3 sequences is further elucidated by analyzing Figures 5.5a, 5.5b, and 5.5c. It becomes evident that the red curve consistently maintains elevated values across all series transformations, not confined solely to the occurrence of change points. This characteristic poses a challenge in accurately identifying the peaks.

Analyzing Sequence 4 AUC results, it becomes evident that the outcomes of SAXJS variations surpass those of Sequences 1, 2, and 3, Specifically, SAXJS-T, SAXJS-$\tau$, SAX-BOP and RuLSIF exhibit AUC values exceeding 0.7. Observing the Sequence 4 (Figure 5.5d), the Jensen-Shannon distance behavior of keeping high values during all the

Figure 5.5: (Bee dance data Sequence 1 to 6. Illustrative time series samples (upper) and the best change point score obtained by our proposed method (lower). Black vertical lines mark the true change points.



(a) Sequence 1

(b) Sequence 2

(c) Sequence 3

(d) Sequence 4

(e) Sequence 5

(f) Sequence 6

Source: Created by the author.

series is also observed in this Sequence, but less predominant than in Sequence 1 and 2. It also seems to have many false positive peaks.

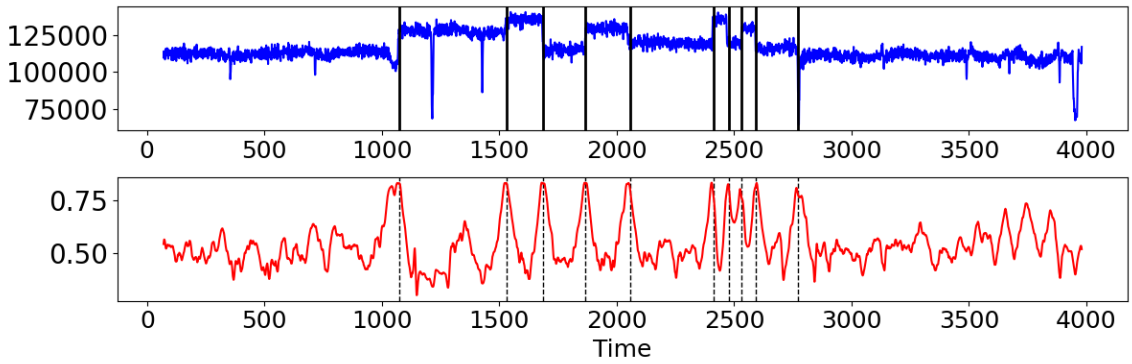
For Sequence 5, the SAXJS-$\tau$ and SAXJS-BOP have satisfactorily AUC values higher than 0.80, and no other CPD method has similar performance. Analyzing Figure 5.5e, we can observe the behavior of high values in some parts of the series, but overall, it is possible to observe that most of the peaks match the real CPs. For Sequence 6, RuLSIF has the bigger AUC of of all state-of-the-art methods, its AUC is 0.857, and all SAXJS variations have a bigger AUC than 0.90, except for SAXJS-BOP. SAXJS-$\tau$ obtains a score of 1.0. In the Sequence 6 (Figure 5.5f), it is possible to notice that peaks match the real CPs, and the peaks that do not match are inside the interval of tolerance.

In conclusion, SAXJS' AUC scores are comparable to other CPD methods, having bad performance in Sequences 1 and 2. In Sequences 4, 5, and 6, SAXJS' and RuLSIF's scores improve, while the other methods, except for KernelCPD in Sequence 5, remain below 0.7. SAXJS-$\tau$ obtains the best score in Sequences 1,2,5 and 6. SAXJS-BOP obtained the best AUC in Sequence 4, and RuLSIF had the best result in Sequence 3.

## 5.4 Real-world cases: Well log

In this Section, we evaluate our framework with the third real-world case called Well log. Well log data set is a representation of a nuclear magnetic response data and changes in the time series' mean correlate with alterations in rock stratification. These time series also have some noisy data represented as anomalies with abrupted low values. As seen in Table 5.8, all SAXJS variations have higher results than other state-of-the-art CPD methods, showing its robustness to noisy data. The lowest AUC score for an SAXJS version was 0.848, and the highest AUC score from other CPD methods was 0.799. These satisfactory results can be seen in Figure 5.6.

Figure 5.6: (Well log data). Illustrative time series samples (upper) and the best change point score obtained by our proposed method (lower). Black vertical lines mark the true change-points.




Source: Created by the author.

Table 5.8: AUC values for Well log using $I = 10$.

| | Well log |
|---|---|
| KernelCPD | 0.778 |
| RuLSIF | **0.799** |
| TIRE | 0.541 |
| BOCPDMS | 0.710 |
| SAXJS-S | **1.0** |
| SAXJS-T | 0.848 |
| SAXJS-$\tau$ | **1.0** |
| SAXJS-BOP | 0.917 |

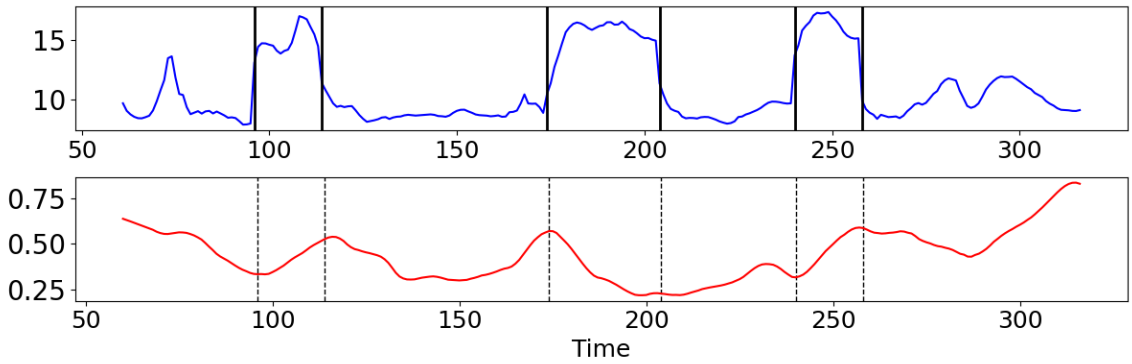
It shows that all points of real CPs (vertical black lines) correspond to peaks in the Jensen-Shannon distance series (red curves). Although we can observe other peaks in the series of Jensen-Shannon values, the peaks corresponding to real CPs are the only ones that exceed the threshold of 0.75.

Observing the Well log data shape in Figure 5.6, it becomes evident that the predominant alterations are in amplitude rather than frequency. This observation could explain why SAXJS-S has the highest result of AUC, along with SAXJS-$\tau$, as SAXJS-S is not adapted to capture shifts in frequency satisfactorily.

## 5.5 Real-world cases: Run log

In this Section, we discuss about the last analyzed real-world case: the Run log data. Run log set represents the pace of a runner during a training session. This data set is the smallest time series from the Real-World and synthetic cases, and it has the worst SAXJS performance of all series as observed in Table 5.9. These results may be due to the difficulty of SAXJS methods dealing with fewer points. However, despite SAXJS-BOP being the version that needed the largest data length, its result was the second highest among all SAXJS variations. Although the method did not stand out, its AUC results were comparable to other CPD methods. In Figure 5.7, we can see that some real CPs match with Jensen-Shannon peaks, and others do not.

Figure 5.7: (Run log data). Illustrative time series samples (upper) and the best change point score obtained by our proposed method (lower). Black vertical lines mark the true change-points.




Source: Created by the author.

Table 5.9: AUC values for Run log using $I = 5$.

| | Run Log |
|---|---|
| KernelCPD | **0.944** |
| RuLSIF | 0.750 |
| TIRE | 0.713 |
| BOCPDMS | 0.867 |
| SAXJS-S | 0.746 |
| SAXJS-T | 0.758 |
| SAXJS-$\tau$ | **0.786** |
| SAXJS-BOP | 0.784 |

## 5.6 Syntethic case: no change points

In this section, we analyze the performance of our Framework in a change point free environment. To conduct this analysis, we retrieve the optimal parameters from the Grid Search for each SAXJS version and other datasets. Subsequently, we execute the best parameter configuration with the minimal Threshold of 0.30 on the time series devoid of change points according to the data set size. The parameter configurations of data sets with bigger length such as JM 1000, SV 1000, CC 1000, all HASC Challenge 2011 participantes were tested in the time series with no change points of size 50000. In contrast, the parameter configurations of data sets with smaller length such as JM 100, SV 100, CC 100, all Bee dance sequences, Well Log and Run Log were tested in the time series free with change points of size 5000. In order to evaluate the results, we use the False Positive Rate(FPR) described in [3] which calculates the rate of False Positive in comparison to all the points in the data set.

In Table 5.10, the outcomes of the optimal configurations for SAXJS-S and SAXJS-T are presented, while Table 5.11 showcases the results corresponding to the best setups of SAXJS-$\tau$ and SAXJS-BOP. Comparing all the SAXJS versions, we can conclude that SAXJS-S has the most optimal results with 14 FPR results being equal to 0.000 (highlighted in blue) and no FPR larger than 0.050 (highlighted in red) which means a more than 5% of the data being considerate False Positive. The second best result was SAXJS-$\tau$ with 8 FPR with 0.000 as a result and 3 above 0.050. The third was SAXJS-T with 6 FPR equal to 0.00 and 3 superior to 0.050. The worst result was SAXJS-BOP with 1 FPR equal to 0.000 and 3 larger than 0.050.

The superior performance of SAXJS-S can be attributed to its utilization of less symbols when comparate to other SAXJS versions. SAXJS-S utilizes only $b$ symbols, which is the number of possible SAX symbols, while SAXJS-T and SAXJS-$\tau$ use $b^2$ symbols and SAXJS-BOP considers $b^d$ being $d$ the number of neighbors to concatenate into a symbol. By employing fewer symbols, less data is required to achieve the objective of avoiding a

Table 5.10: (SAXJS-S and SAXJS-T) False Positive Rate(FPR) for all the data sets with its best parameters configurations previously set with past experiments using Grid Search.

| | SAXJS-S | | SAXJS-T | |
|---|---|---|---|---|
| Data set | Best configuration | FPR | Best configuration | FPR |
| JM 100 | $W = 60, b = 9, g = 31, p = 15$ | 0.000 | $W = 50, b = 3, g = 31, p = 15$ | 0.004 |
| SV 100 | $W = 90, b = 9, g = 11, p = 10$ | 0.000 | $W = 100, b = 7, g = 31, p = 15$ | 0.020 |
| CC 100 | $W = 90, b = 8, g = 41, p = 15$ | 0.000 | $W = 90, b = 9, g = 41, p = 15$ | 0.017 |
| JM 1000 | $W = 300, b = 8, g = 101, p = 20$ | 0.000 | $W = 500, b = 9, g = 201, p = 20$ | 0.000 |
| SV 1000 | $W = 300, b = 9, g = 101, p = 15$ | 0.000 | $W = 500, b = 9, g = 201, p = 20$ | 0.000 |
| CC 1000 | $W = 900, b = 8, g = 201, p = 15$ | 0.000 | $W = 900, b = 9, g = 201, p = 15$ | 0.000 |
| HASC 2011 p101 | $W = 300, b = 5, g = 201, p = 20$ | 0.000 | $W = 500, b = 4, g = 201, p = 5$ | 0.000 |
| HASC 2011 p102 | $W = 300, b = 5, g = 201, p = 20$ | 0.000 | $W = 200, b = 9, g = 201, p = 20$ | 0.006 |
| HASC 2011 p103 | $W = 200, b = 5, g = 81, p = 15$ | 0.000 | $W = 500, b = 9, g = 201, p = 5$ | 0.000 |
| HASC 2011 p104 | $W = 300, b = 5, g = 201, p = 20$ | 0.000 | $W = 200, b = 7, g = 201, p = 10$ | 0.001 |
| HASC 2011 p105 | $W = 300, b = 5, g = 201, p = 20$ | 0.000 | $W = 300, b = 5, g = 201, p = 20$ | 0.000 |
| Bee dance Seq1 | $W = 30, b = 5, g = 5, p = 15$ | 0.005 | $W = 30, b = 6, g = 5, p = 5$ | 0.068 |
| Bee dance Seq2 | $W = 30, b = 7, g = 15, p = 5$ | 0.012 | $W = 40, b = 5, g = 5, p = 5$ | 0.065 |
| Bee dance Seq3 | $W = 60, b = 4, g = 5, p = 15$ | 0.000 | $W = 90, b = 6, g = 5, p = 20$ | 0.025 |
| Bee dance Seq4 | $W = 30, b = 9, g = 5, p = 5$ | 0.033 | $W = 40, b = 4, g = 5, p = 5$ | 0.057 |
| Bee dance Seq5 | $W = 30, b = 5, g = 5, p = 5$ | 0.005 | $W = 50, b = 4, g = 5, p = 5$ | 0.040 |
| Bee dance Seq6 | $W = 40, b = 3, g = 5, p = 5$ | 0.000 | $W = 40, b = 3, g = 15, p = 5$ | 0.014 |
| Run log | $W = 30, b = 4, g = 11, p = 5$ | 0.002 | $W = 30, b = 6, g = 15, p = 10$ | 0.035 |
| Well log | $W = 60, b = 3, g = 25, p = 5$ | 0.000 | $W = 60, b = 3, g = 15, p = 20$ | 0.003 |

sparse probability distribution vector and prevents the calculation of Jensen-Shannon of showing anomalous results. This hypotheses of less symbols is confirmed by observing the both tables and analysing that instances of 0.000 in FPR for SAXJS-S, absent in SAXJS-$\tau$,

Table 5.11: (SAXJS-$\tau$ and SAXJS-BOP) False Positive Rate(FPR) for all the data sets with its best parameters configurations previously set with past experiments using Grid Search.

| | SAXJS-$\tau$ | | SAXJS-BOP | |
|---|---|---|---|---|
| Data set | Best configuration | FPR | Best configuration | FPR |
| JM 100 | $W = 60, b = 4, \tau = 11, g = 41, p = 15$ | 0.013 | $W = 70, d = 2, b = 4, g = 41, p = 15$ | 0.017 |
| SV 100 | $W = 100, b = 5, \tau = 7, g = 21, p = 15$ | 0.008 | $W = 100, d = 2, b = 4, g = 41, p = 15$ | 0.014 |
| CC 100 | $W = 50, b = 9, \tau = 10, g = 11, p = 5$ | 0.056 | $W = 50, d = 3, b = 7, g = 11, p = 5$ | 0.056 |
| JM 1000 | $W = 300, b = 8, \tau = 3, g = 201, p = 20$ | 0.000 | $W = 400, d = 3, b = 4, g = 201, p = 10$ | 0.006 |
| SV 1000 | $W = 300, b = 9, \tau = 14, g = 201, p = 20$ | 0.000 | $W = 700, d = 3, b = 5, g = 201, p = 20$ | 0.005 |
| CC 1000 | $W = 500, b = 9, \tau = 2, g = 201, p = 20$ | 0.000 | $W = 500, d = 2, b = 7, g = 201, p = 20$ | 0.000 |
| HASC 2011 p101 | $W = 600, b = 7, \tau = 11, g = 201, p = 5$ | 0.000 | $W = 600, d = 3, b = 7, g = 201, p = 20$ | 0.005 |
| HASC 2011 p102 | $W = 400, b = 6, \tau = 6, g = 101, p = 10$ | 0.000 | $W = 200, d = 3, b = 4, g = 201, p = 15$ | 0.006 |
| HASC 2011 p103 | $W = 500, b = 8, \tau = 8, g = 201, p = 5$ | 0.000 | $W = 500, d = 3, b = 6, g = 201, p = 20$ | 0.005 |
| HASC 2011 p104 | $W = 600, b = 9, \tau = 14, g = 201, p = 10$ | 0.000 | $W = 600, d = 3, b = 7, g = 201, p = 10$ | 0.007 |
| HASC 2011 p105 | $W = 500, b = 7, \tau = 7, g = 201, p = 20$ | 0.000 | $W = 600, d = 3, b = 7, g = 201, p = 10$ | 0.007 |
| Bee dance Seq1 | $W = 60, b = 5, \tau = 10, g = 15, p = 15$ | 0.021 | $W = 30, d = 3, b = 7, g = 5, p = 5$ | 0.052 |
| Bee dance Seq2 | $W = 50, b = 6, \tau = 10, g = 5, p = 5$ | 0.068 | $W = 30, d = 3, b = 7, g = 5, p = 5$ | 0.052 |
| Bee dance Seq3 | $W = 70, b = 4, \tau = 10, g = 11, p = 11$ | 0.012 | $W = 100, d = 3, b = 4, g = 11, p = 15$ | 0.021 |
| Bee dance Seq4 | $W = 80, b = 7, \tau = 4, g = 5, p = 15$ | 0.024 | $W = 30, d = 3, b = 4, g = 11, p = 15$ | 0.028 |
| Bee dance Seq5 | $W = 40, b = 7, \tau = 10, g = 11, p = 5$ | 0.059 | $W = 30, d = 3, b = 7, g = 5, p = 10$ | 0.022 |
| Bee dance Seq6 | $W = 50, b = 7, \tau = 9, g = 11, p = 10$ | 0.032 | $W = 40, d = 2, b = 3, g = 11, p = 10$ | 0.033 |
| Run log | $W = 60, b = 4, \tau = 4, g = 15, p = 15$ | 0.018 | $W = 30, d = 2, b = 2, g = 11, p = 15$ | 0.014 |
| Well log | $W = 70, b = 7, \tau = 7, g = 15, p = 20$ | 0.019 | $W = 60, d = 3, b = 2, g = 25, p = 20$ | 0.017 |

occurred solely within smaller datasets where the window size (W) was either smaller or equal to 100.

The SAXJS-T and SAXJS-$\tau$ were very similar with a difference of only 2 FPR with value 0.000 in SAXJS-$\tau$. In conclusion, all versions of SAXJS yielded satisfactory results, with numerous instances of FPR equal to 0.000 and only a few values equal to or exceeding 0.050, except for SAXJS-BOP, which had only one instance of 0.000 FPR. SAXJS-BOP bad performance can be explained as it needs a larger window size(W) in comparison to others, which reinforces its necessity of more data to work properly.

## 5.7 Final Remarks

This chapter presents the results of all SAXJS variations and other CPD methods in diverse scenarios, including synthetic data sets and real-world cases with different characteristics, such as length, number of CPs, and CPs distances. Analyzing the outcomes of CPD in all these real-world cases, it is evident that all variations of SAXJS have adapted well to diverse contexts.

SAXJS has the largest AUC for all synthetic and real-world data, except for Run log and HASC with a tolerance interval of $I = 100$. Despite not having the best AUC result, the AUC values of all SAXJS variations are satisfactory and above 0.7 in both cases. In general, SAXJS has the best AUC result in 5 out of 7 analysed cases (in analyzed data, we include synthetic data with changes every 100 and 1000, HASC with interval tolerance of $I = 10$ and $I = 100$, Bee dance, Well log, Run log), approximately 71% of success rate.

All SAXJS variations have very similar results, although SAXJS-S when compared to other SAXJS options, has the worst performance in 5 out of 8 analyzed context (including the synthetic data without change points inserted). SAXJS-S only achieves its best results compared to other SAXJS versions in the environment free of change points and the Well Log context, where there is no frequency shift, which evidences its difficulty in capturing this type of change. The difficulty SAXJS-S faces in capturing frequencies becomes more apparent when analyzing a synthetic data set. In this scenario, its AUC demonstrates a less favorable performance compared to most other CPD methods when assessing coefficient changes.

Consequently, we deduce that constructing a probability distribution based on the connections between SAX symbols, and the different aspects between SAXJS-S and SAXJS-T versions reveals positive outcomes. Although SAXJS-$\tau$ has the largest AUC results compared to SAXJS-T, the largest AUC difference between the methods is 0.152 in Well log data. The performance of SAXJS-$\tau$ on time series without change points also surpassed that of SAXJS-T, albeit by a narrow margin of only 2 FPR values at 0.000. SAXJS-BOP also shows satisfactory AUC even in small data sets such as Run log and Bee

dance, which could be a withdrawal of the method. However, SAXJS-BOP had the worst and unsatisfactory result in the analysis of the scenario with the absence of change points.

# Chapter 6

# Conclusion and Future Directions

This Dissertation proposes a framework to detect change points in univariate time series. It is an important field with several applications in different contexts, such as climate change, human activity detection, health monitoring, and image feature selection. There are decades of study in change point detection, still, the necessity of online, low computational cost and adaptable to multiple systems algorithms remains as a solution to deal with streaming large data.

The framework comprises five steps: SAX transformation, probability distribution calculation, Jensen-Shannon distance calculation, and Data smoothing and peak detection. As Data smoothing and peak detection steps add more points to the sliding window, in Future Work, we plan to investigate new techniques to detect the peaks using its prominence information and then reduce the number of points ahead of time $t$ needed to detect the change point. Consequently, by reducing the number of points ahead of $t$, we ensure that the algorithm is closer to real-time.

The probability distribution calculation is the most important step as we introduce four alternatives. The first option corresponds to the SAXJS-S version and consists of calculating the probability distribution based on the frequency of SAX symbols. The second option corresponds to SAXJS-T and SAXJS-$\tau$. In these variations, we introduce a new data representation based on a graph construction of the SAX connections. The only difference between SAXJS-T and SAXJS-$\tau$ is using a variable defined by the user called $\tau$, which is the jump in connections to construct the graph. The last option is to use a variation of BOP introduced in our work in which the sliding window division is made before the SAX transformation to keep its online characteristic.

We evaluate our framework's ability to detect change points with a synthetic data set with mean, variance, and coefficient changes every 100 and 1000 steps and other real-world cases. In choosing real-world cases, we focus on benchmark data for change point analysis from different contexts with different lengths and change point distances. Our proposed algorithm has the most significant AUC (evaluation metric we use) compared to other change point algorithms in most synthetic and real-world data, proving that our framework competes with the literature and even outperforms it. After evaluating our framework in a synthetic environment with changes in mean, variance and coefficient and

in real-world scenarios, we proceeded to conduct an experiment with synthetic datasets devoid of any inserted change points. In this experiment, SAXJS-S had the best result and SAXJS-$\tau$ and SAXJS-T also demonstrate satisfactory outcomes. SAXJS-BOP was the only SAXJS versions which exhibited poor performance.

In addition to the satisfactory results in AUC comparison, our framework has other advantages when compared to different change point algorithms we tested, which are its online, robustness to noise, and low computational cost nature. We conducted tests using Change Point Detection (CPD) algorithms employing diverse strategies. The algorithms selected for comparison with our framework include TIRE, KernelCPD, RuLSIF, BOCPDMS, and ADAGA.

TIRE relies on a Subspace Model using autoencoder, KernelCPD employs a Kernel-Based method, RuLSIF utilizes a Likelihood Ratio strategy based on estimating density ratio, and BOCPDMS operates as a probabilistic method with model selection. KernelCPD and TIRE are offline, and RuLSIF has been adapted for online use in this study. BOCPDMS is an online method but is parametric. We tried to test our data sets with ADAGA, a nonparametric and online application, although we had a problem with runtime, and the analysis was left for future work. SAXJS has another advantage in comparison to RuLSIF which is it does not have a cross-validation step that may burden the process.

As Future work, we also plan to do a sensitivity analysis of our algorithm to test its robustness to different hyperparameter configurations and use statistical tests to define the best threshold options. In addition, we plan to construct an open source library of our framework and adapt our algorithm for multivariate context by using other dimension information in SAX transformation.

Previous to this work, in [55], we presented some preliminary results of this dissertation about the use of SAXJS-$\tau$ in the context of wearable sensor data.

# Bibliography

[1] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[2] Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[3] Samaneh Aminikhanghahi, Tinghui Wang, and Diane J Cook. Real-time change point detection with application to smart home time series data. *IEEE Transactions on Knowledge and Data Engineering*, 31(5):1010–1023, 2018.

[4] Sylvain Arlot, Alain Celisse, and Zaid Harchaoui. A kernel multiple change-point algorithm via model selection. *Journal of machine learning research*, 20(162), 2019.

[5] Michele Basseville, Igor V Nikiforov, et al. *Detection of abrupt changes: theory and application*, volume 104. prentice Hall Englewood Cliffs, 1993.

[6] Ayse S Cakmak, Giulia Da Poian, Adam Willats, Ammer Haffar, Rami Abdulbaki, Yi-An Ko, Amit J Shah, Viola Vaccarino, Donald L Bliwise, Christopher Rozell, et al. An unbiased, efficient sleep–wake detection algorithm for a population with sleep disorders: change point decoder. *Sleep*, 43(8):zsaa011, 2020.

[7] Edoardo Caldarelli, Philippe Wenk, Stefan Bauer, and Andreas Krause. Adaptive gaussian process change point detection. In *International Conference on Machine Learning*, pages 2542–2571. PMLR, 2022.

[8] Alain Celisse, Guillemette Marot, Morgane Pierre-Jean, and GJ Rigaill. New efficient algorithms for multiple change-point detection with reproducing kernels. *Computational Statistics & Data Analysis*, 128:200–220, 2018.

[9] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models. *arXiv preprint arXiv:1901.06077*, 2019.

[10] Hao Chen and Nancy Zhang. Graph-based change-point detection. *The Annals of Statistics*, 43(1):139–176, 2015.

[11] Kevin C Cheng, Eric L Miller, Michael C Hughes, and Shuchin Aeron. On matched filtering for statistical change point detection. *IEEE Open Journal of Signal Processing*, 1:159–176, 2020.

[12] Ian Cleland, Manhyung Han, Chris Nugent, Hosung Lee, Sally McClean, Shuai Zhang, and Sungyoung Lee. Evaluation of prompted annotation of activity data recorded from a smart phone. *Sensors*, 14(9):15861–15879, 2014.

[13] Arne De Brabandere, Zhenxiang Cao, Maarten De Vos, Alexander Bertrand, and Jesse Davis. Semi-supervised change point detection using active learning. In *International Conference on Discovery Science*, pages 74–88. Springer, 2022.

[14] Tim De Ryck, Maarten De Vos, and Alexander Bertrand. Change point detection in time series data using autoencoders with a time-invariant representation. *IEEE Transactions on Signal Processing*, 69:3513–3524, 2021.

[15] Zahra Ebrahimzadeh, Min Zheng, Selcuk Karakas, and Samantha Kleinberg. Pyramid recurrent neural networks for multi-scale change-point detection. 2018.

[16] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34, 2012.

[17] Amin Fakhrazari and Hamid Vakilzadian. A survey on time series data mining. In *2017 IEEE International Conference on Electro Information Technology (EIT)*, pages 476–481. IEEE, 2017.

[18] Wei Fan, Kun Zhang, Hong Cheng, Jing Gao, Xifeng Yan, Jiawei Han, Philip Yu, and Olivier Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 230–238, 2008.

[19] Johann Faouzi. Time series classification: A review of algorithms and implementations. *Machine Learning (Emerging Trends and Applications)*, 2022.

[20] Len Feremans, Boris Cule, and Bart Goethals. Petsc: pattern-based embedding for time series classification. *Data Mining and Knowledge Discovery*, 36(3):1015–1061, 2022.

[21] Kyle D Feuz, Diane J Cook, Cody Rosasco, Kayela Robertson, and Maureen Schmitter-Edgecombe. Automated detection of activity transitions for prompting. *IEEE transactions on human-machine systems*, 45(5):575–585, 2014.

[22] Dmitriy Fradkin and Fabian Mörchen. Mining sequential patterns for classification. *Knowledge and Information Systems*, 45:731–749, 2015.

[23] Ramtin Hamavar and Babak Mohammadzadeh Asl. Seizure onset detection based on detection of changes in brain activity quantified by evolutionary game theory model. *Computer Methods and Programs in Biomedicine*, 199:105899, 2021.

[24] Manhyung Han, La The Vinh, Young-Koo Lee, and Sungyoung Lee. Comprehensive context recognizer based on multimodal sensors in a smartphone. *Sensors*, 12(9):12588–12605, 2012.

[25] Zhenwen He, Shirong Long, Xiaogang Ma, and Hong Zhao. A boundary distance-based symbolic aggregate approximation method for time series data. *Algorithms*, 13(11):284, 2020.

[26] Shohei Hido, Tsuyoshi Idé, Hisashi Kashima, Harunobu Kubo, and Hirofumi Matsuzawa. Unsupervised change analysis using supervised learning. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 148–159. Springer, 2008.

[27] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445, 2009.

[28] Nobuo Kawaguchi, Nobuhiro Ogawa, Yohei Iwasaki, Katsuhiko Kaji, Tsutomu Terada, Kazuya Murao, Sozo Inoue, Yoshihiro Kawahara, Yasuyuki Sumi, and Nobuhiko Nishio. Hasc challenge: gathering large scale human activity corpus for the real-world activity understandings. In *Proceedings of the 2nd augmented human international conference*, pages 1–5, 2011.

[29] Nobuo Kawaguchi, Ying Yang, Tianhui Yang, Nobuhiro Ogawa, Yohei Iwasaki, Katsuhiko Kaji, Tsutomu Terada, Kazuya Murao, Sozo Inoue, Yoshihiro Kawahara, et al. Hasc2011corpus: towards the common ground of human activity recognition. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 571–572, 2011.

[30] Yoshinobu Kawahara and Masashi Sugiyama. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 389–400. SIAM, 2009.

[31] Yoshinobu Kawahara, Takehisa Yairi, and Kazuo Machida. Change-point detection in time-series data based on subspace identification. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 559–564. IEEE, 2007.

[32] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[33] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Proceedings 2001 IEEE international conference on data mining*, pages 289–296. IEEE, 2001.

[34] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[35] Jeremias Knoblauch and Theodoros Damoulas. Spatio-temporal bayesian on-line changepoint detection with model selection. In *International Conference on Machine Learning*, pages 2718–2727. PMLR, 2018.

[36] Jeremias Knoblauch, Jack E Jewson, and Theodoros Damoulas. Doubly robust bayesian inference for non-stationary streaming data with *beta*-divergences. *Advances in Neural Information Processing Systems*, 31, 2018.

[37] James Large, Anthony Bagnall, Simon Malinowski, and Romain Tavenard. On time series classification with dictionary-based classifiers. *Intelligent Data Analysis*, 23(5):1073–1089, 2019.

[38] RJ Larzen and ML Marx. An introduction to mathematical statistics and it's applications, 1981.

[39] Wei-Han Lee, Jorge Ortiz, Bongjun Ko, and Ruby Lee. Time series segmentation through automatic feature learning. *arXiv preprint arXiv:1801.05394*, 2018.

[40] Yangtao Li, Tengfei Bao, Xiaosong Shu, Zhixin Gao, Jian Gong, and Kang Zhang. Data-driven crack behavior anomaly identification method for concrete dams in long-term service using offline and online change point detection. *Journal of Civil Structural Health Monitoring*, 11(5):1449–1460, 2021.

[41] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, 2003.

[42] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[43] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, 39:287–315, 2012.

[44] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[45] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[46] Battuguldur Lkhagva, Yu Suzuki, and Kyoji Kawagoe. New time series data representation esax for financial applications. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pages x115–x115. IEEE, 2006.

[47] Matthew Middlehurst, William Vickers, and Anthony Bagnall. Scalable dictionary classifiers for time series classification. In *Intelligent Data Engineering and Automated Learning–IDEAL 2019: 20th International Conference, Manchester, UK, November 14–16, 2019, Proceedings, Part I 20*, pages 11–19. Springer, 2019.

[48] Valentina Moskvina and Anatoly Zhigljavsky. An algorithm based on singular spectrum analysis for change-point detection. *Communications in Statistics-Simulation and Computation*, 32(2):319–352, 2003.

[49] Attila M Nagy and Vilmos Simon. A novel data representation method for smart cities' big data. *Artificial Intelligence, Machine Learning, and Optimization Tools for Smart Cities: Designing for Sustainability*, pages 97–122, 2022.

[50] Sang Min Oh, James M Rehg, Tucker Balch, and Frank Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *International Journal of Computer Vision*, 77:103–124, 2008.

[51] ES Page. A test for a change in a parameter occurring at an unknown point. *Biometrika*, 42(3/4):523–527, 1955.

[52] William H Press and Saul A Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672, 1990.

[53] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):1–27, 2010.

[54] J Larson Richard, LM Morris, et al. An introduction to mathematical statistics and its applications, 2000.

[55] Giovanna A Riqueti, Pedro H Barros, João B Borges, Felipe D Cunha, Osvaldo A Rosso, and Heitor S Ramos. Saxjs: An online change point detection for wearable sensor data. In *Anais do XLI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 351–364. SBC, 2023.

[56] Oè Ruanaidh. Jjk and fitzgerald, wj (1996) numerical bayesian methods applied to signal processing. *Statistics and Computing. Springer New York*, 2000.

[57] Patrick Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29:1505–1530, 2015.

[58] Patrick Schäfer and Mikael Högqvist. Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th international conference on extending database technology*, pages 516–527, 2012.

[59] Patrick Schäfer and Ulf Leser. Fast and accurate time series classification with weasel. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 637–646, 2017.

[60] Patrick Schäfer and Ulf Leser. Multivariate time series classification with weasel+ muse. *arXiv preprint arXiv:1711.11343*, 2017.

[61] Pavel Senin and Sergey Malinchik. Sax-vsm: Interpretable time series classification using sax and vector space model. In *2013 IEEE 13th international conference on data mining*, pages 1175–1180. IEEE, 2013.

[62] Chetan Sharma and CSP Ojha. Modified signal-to-noise ratio method for early detection of climate change. *Journal of Hydrologic Engineering*, 25(8):04020032, 2020.

[63] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[64] Youqiang Sun, Jiuyong Li, Jixue Liu, Bingyu Sun, and Christopher Chow. An improvement of symbolic aggregate approximation distance measure for time series. *Neurocomputing*, 138:189–198, 2014.

[65] Jun-ichi Takeuchi and Kenji Yamanishi. A unifying framework for detecting outliers and change points from time series. *IEEE transactions on Knowledge and Data Engineering*, 18(4):482–492, 2006.

[66] Francisco Traversaro, Francisco O Redelico, Marcelo R Risk, Alejandro C Frery, and Osvaldo A Rosso. Bandt-pompe symbolization dynamics for time series with tied values: A data-driven approach. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(7):075502, 2018.

[67] Charles Truong, Laurent Oudre, and Nicolas Vayatis. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020.

[68] Ryan Darby Turner. *Gaussian processes for state space models and change point detection*. PhD thesis, University of Cambridge, 2012.

[69] Gerrit JJ Van den Burg and Christopher KI Williams. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*, 2020.

[70] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[71] Shmuel Winograd. On computing the discrete fourier transform. *Mathematics of computation*, 32(141):175–199, 1978.

[72] Xiang Xuan and Kevin Murphy. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th international conference on Machine learning*, pages 1055–1062, 2007.

[73] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370, 2013.

[74] Kenji Yamanishi and Jun-ichi Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 676–681, 2002.

[75] Jining Yan, Lizhe Wang, Weijing Song, Yunliang Chen, Xiaodao Chen, and Ze Deng. A time-series classification approach based on change detection for rapid land cover mapping. *ISPRS Journal of Photogrammetry and Remote Sensing*, 158:249–262, 2019.

[76] Lijuan Yan, Xiaotao Wu, and Jiaqing Xiao. An improved time series symbolic representation based on multiple features and vector frequency difference. *Journal of Computer and Communications*, 10(06):44–62, 2022.

[77] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. Clustering time series using unsupervised-shapelets. In *2012 IEEE 12th International Conference on Data Mining*, pages 785–794. IEEE, 2012.

[78] Chaw Thet Zan and Hayato Yamana. An improved symbolic aggregate approximation distance measure based on its statistical features. In *Proceedings of the 18th*

*international conference on information integration and web-based applications and services*, pages 72–80, 2016.

[79] Haowen Zhang, Yabo Dong, and Duanqing Xu. Entropy-based symbolic aggregate approximation representation method for time series. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 905–909. IEEE, 2020.

[80] Luciano Zunino, Felipe Olivares, Haroldo V Ribeiro, and Osvaldo A Rosso. Permutation jensen-shannon distance: A versatile and fast symbolic tool for complex time-series analysis. *Physical Review E*, 105(4):045310, 2022.

# Appendix A

# ADAGA parameter configurations and Experiments results

In this chapter, we outline the hyperparameters intended for utilization in grid search for ADAGA, resenting their respective configurations for every dataset under examination. Subsequently, we provide the results of the ADAGA that were successfully executed.

ADAGA uses the following hyperparameters:

- kernel-approx: kernel approximation strategy. It has three options to approximate the Gaussian Kernel (inducing points, QFF, and linear approximation).
- n: number of points to use in the approximation strategy.
- Kernel: kernel model.
- min-window-size: Minimum number of points in a window.
- n-batchs: number of batches in which the overall trajectory is partitioned.
- delta: the delta hyperparameter to calculate the thresholds.

We planned to use the batch size as 1 for all scenarios to obtain more accurate results, as suggested in [7]. We would test all the kernel possibilities: Radial Basis Function (RBF), Rational Quadratic (RQ), Matern52, Linear, and Periodic, and all three possible methods for calculating Kernel approximation. The hyperparameters change according to the length of the series and the Kernel approximation approach. For each data set, we use the following hyperparameter configuration:

- **Synthetic cases (interval 100)**:
  - For the inducing points strategy:
    - n:(10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210)
    - min-window-size: (15, 45, 75, 115, 145, 175, 205, 235, 265, 295)
    - delta: (0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
  - For the QFFs strategy:

    - n:(10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210)
    - min-window-size: (15, 45, 75, 115, 145, 175, 205, 235, 265, 295)
    - delta: (0.05, 0.1, 0.2, 0.3, 0.4)
  - For the Linear approximation strategy:
    - min-window-size: (15, 45, 75, 115, 145, 175, 205, 235, 265, 295)
    - delta: (0.05, 0.1, 0.2, 0.3, 0.4)

- **Synthetic cases (interval 1000) and real-world case: HASC Challenge 2011**:
  - For the inducing points strategy:
    - n:(10, 50, 100, 200, 300, 400, 700, 800)
    - min-window-size: (15, 55, 115, 215, 315, 415, 515, 615)
    - delta: (0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
  - For the QFFs strategy:
    - n:(10, 50, 100, 200, 300, 400, 700, 800)
    - min-window-size: (15, 55, 115, 215, 315, 415, 515, 615)
    - delta: (0.05, 0.1, 0.2, 0.3, 0.4)
  - For the Linear approximation strategy:
    - min-window-size: (15, 55, 115, 215, 315, 415, 515, 615, 715, 815)
    - delta: (0.05, 0.1, 0.2, 0.3, 0.4)

- **Real-world cases: Bee dance and Run log**:
  - For the inducing points strategy:
    - n:(10, 30, 50, 70, 90, 110)
    - min-window-size: (15, 35, 55, 75, 95, 115)
    - delta: (0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
  - For the QFFs strategy:
    - n:(10, 30, 50, 70, 90, 110)
    - min-window-size: (15, 35, 55, 75, 95, 115)
    - delta: (0.05, 0.1, 0.2, 0.3, 0.4)
  - For the Linear approximation strategy:
    - min-window-size: (15, 35, 55, 75, 95, 115, 135, 145)
    - delta: (0.05, 0.1, 0.2, 0.3, 0.4)

- **Real-world cases: Well log**:

- For the inducing points strategy:
  - n:(10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210)
  - min-window-size: (15, 45, 75, 115, 145, 175, 205, 235)
  - delta: (0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
- For the QFFs strategy:
  - n:(10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210)
  - min-window-size: (15, 45, 75, 115, 145, 175, 205, 235)
  - delta: (0.05, 0.1, 0.2, 0.3, 0.4)
- For the Linear approximation strategy:
  - min-window-size: (15, 45, 75, 115, 145, 175, 205, 235, 265, 295)
  - delta: (0.05, 0.1, 0.2, 0.3, 0.4)

As the Run log is the smallest time series we tested, we could run the grid search just for it without exceeding the 5-day limit. The result for the Run log was an AUC of 0.767, very similar to other algorithms whose AUC values varied between 0.944 and 0.713. We could test ADAGA default hyperparameter configuration with the synthetic data set (changes every 100 steps), and the results are unsatisfactory, as we can observe in Table A.1.

Table A.1: AUC values for synthetic datasets with changes every 100 steps

| | JM 100 | SV 100 | CC 100 | Average AUC |
|---|---|---|---|---|
| ADAGA | 0.275 | 0.402 | 0.405 | 0.361 |

# Appendix B

# Detection delay tables

In this chapter, we present the detection delay information for all SAXJS variations and state-of-the-art methods for all data sets, except those with an interval tolerance greater than 50. The Tables B.1, B.2, B.3, B.4 and B.5 shows the results for synthetic data sets with changes every 100 steps, HASC, Bee dance, Well Log and Run log, respectively. Observing all these tables, it is possible to notice that detection delay variation between CPD methods is small, and it does not bring insights into the quality of the methods.

Table B.1: (Synthetic data with changes in every 100 points). Detection delay comparison of each CPD method with best parameters setting according to AUC.

| | JM 100 | SV 100 | CC 100 | Average Delay |
|---|---|---|---|---|
| KernelCPD | 2.93 | 4.58 | 4.75 | 4.09 |
| RulSIF | 3.69 | 4.39 | 3.89 | 3.99 |
| TIRE | **2.88** | 3.57 | 5.04 | 3.83 |
| BOCPDMS | 5.00 | **2.74** | **3.60** | **3.78** |
| SAXJS-S | 3.00 | **2.77** | 5.82 | 3.86 |
| SAXJS-T | 2.71 | 3.38 | **4.03** | 3.37 |
| SAXJS-$\tau$ | **2.54** | 3.20 | 4.12 | **3.29** |
| SAXJS-BOP | 3.29 | 3.34 | 5.14 | 3.92 |

Table B.2: (HASC Challenge 2011 data). Detection delay comparison of each CPD method with best parameters setting according to AUC for tolerance interval $I = 10$.

| | person101 | person102 | person103 | person104 | person105 | Average |
|---|---|---|---|---|---|---|
| KernelCPD | 3.14 | **0.00** | 3.29 | 2.75 | 3.62 | 2.56 |
| RulSIF | 5.83 | 4.75 | 4.86 | 6.33 | 4.57 | 4.39 |
| TIRE | 5.00 | **0.00** | 5.20 | **2.00** | 8.00 | 4.04 |
| BOCPDMS | **2.14** | 2.89 | **1.50** | 3.67 | **2.57** | **2.13** |
| SAXJS-S | 5.12 | 4.71 | 6.00 | 5.62 | 5.00 | 4.41 |
| SAXJS-T | 4.33 | **4.14** | 6.25 | 4.12 | 4.67 | 3.92 |
| SAXJS-$\tau$ | 4.00 | 5.62 | **3.22** | **3.44** | **3.33** | **3.27** |
| SAXJS-BOP | **3.22** | 4.57 | 5.38 | 5.50 | 4.55 | 3.87 |

Table B.3: (Bee dance data). Detection delay comparison of each CPD method with best parameters setting according to AUC.

| | Seq1 | Seq2 | Seq3 | Seq4 | Seq5 | Seq6 | Average |
|---|---|---|---|---|---|---|---|
| KernelCPD | **4.06** | 4.45 | 4.62 | 5.06 | **4.21** | **4.00** | **4.40** |
| RulSIF | 4.73 | 5.33 | 4.38 | 4.79 | 4.65 | 4.42 | 4.72 |
| TIRE | 4.38 | **5.00** | 5.60 | 5.86 | 5.94 | 5.18 | 5.33 |
| BOCPDMS | 6.67 | 6.64 | **4.20** | **4.33** | 5.11 | 4.38 | 5.22 |
| SAXJS-S | 4.40 | **5.00** | 4.86 | 5.00 | 5.06 | 4.46 | 4.80 |
| SAXJS-T | 4.80 | 5.65 | **4.40** | 4.36 | 3.95 | 5.23 | 4.73 |
| SAXJS-$\tau$ | 5.69 | 5.24 | 7.00 | 6.50 | 3.96 | 3.93 | 5.39 |
| SAXJS-BOP | **4.25** | 5.10 | 5.00 | **4.20** | **3.61** | **3.92** | **4.35** |

Table B.4: (Well Log data). Detection delay comparison of each CPD method with best parameters setting according to AUC.

| | Well Log |
|---|---|
| KernelCPD | **2.82** |
| RulSIF | 4.50 |
| TIRE | 4.00 |
| BOCPDMS | 4.00 |
| SAXJS-S | 3.70 |
| SAXJS-T | 3.11 |
| SAXJS-$\tau$ | 5.40 |
| SAXJS-BOP | **2.90** |

Table B.5: (Run Log data). Detection delay comparison of each CPD method with best parameters setting according to AUC.

| | Run Log |
|---|---|
| KernelCPD | **1.25** |
| RulSIF | 2.33 |
| TIRE | 1.40 |
| BOCPDMS | **1.25** |
| ADAGA | 0.75 |
| SAXJS-S | 1.33 |
| SAXJS-T | 1.67 |
| SAXJS-$\tau$ | **1.25** |
| SAXJS-BOP | 2.00 |